НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 519.688

«До захисту допущено»
Завідувач кафедри СПСКС

_____Віталій РОМАНКЕВИЧ

"___"_____2020р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія

на тему: Модифікований спосіб обробки зображень для томографічних знімків_

Виконав: студент II курсу, групи КВ-92мп
Грицаєнко Віктор Павлович _____

Науковий керівник доц., с.н.с., к.т.н. Юлія БОЯРІНОВА _____

Рецензент_____ _____

Консультант з нормоконтролю доцент, с.н.с.,к.т.н. Юлія БОЯРІНОВА _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
за освітньо-професійною програмою
Спеціальність 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«_1__» _грудня__2019р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Грицаєнко Віктор Павлович
(прізвище, ім'я, по батькові)

1. Тема дисертації «Модифікований спосіб обробки зображень для томографічних знімків»,
науковий керівник дисертації с.н.с.,к.т.н. Боярінова Юлія Євгенівна,

затверджені наказом по університету від «12» листопада 2020 р. №3298-С

2. Термін подання студентом дисертації 10 грудня 2020 р.

3. Об'єкт дослідження обробка зображень

4. Предмет дослідження методи порівняння томографічних зображень

5. Перелік завдань, які потрібно розробити аналіз існуючих методів порівняння зображень; обгрунтування доцільності розробки алгоритму для

порівняння медицинських томографічних зображень; модифікація алгоритму Average Hash; розробка алгоритму порівняння зображень блочним методом для томографічних знімків; аналіз розробленого алгоритму та його порівняння з існуючими.

6. Перелік ілюстративного матеріалу - презентація_____

7. Перелік публікацій - 2 тез _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Затвердження теми	1.12.2019	
2	Збір та дослідження літератури	19.09.2020	
3	Аналіз існуючих рішень	23.09.2020	
4	Зміст та вступ	25.09.2020	
5	Розробка програмної моделі	05.10.2020	
6	Реферат	09.10.2020	
7	Перший розділ	09.10.2020	
8	Другий розділ	23.10.2020	
9	Третій розділ	04.11.2020	
10	Четвертий розділ	16.11.2020	
11	Редагування та перевірка роботи	20.11.2020	
12	Попередній розгляд магістерської дисертації на кафедрі	27.11.2020	

Студент _____ Віктор ГРИЦАЄНКО

Науковий керівник дисертації _____ Юлія БОЯРІНОВА__

РЕФЕРАТ

Актуальність теми.

Обробка зображень та комп'ютерний зір є ключовими дисциплінами роботи з зображеннями та використовуються для розробки прикладних програм та систем реального часу для вирішення аналітичних, дослідницьких та наукових задач. Медичні зображення є важливою складовою медичних досліджень та охорони здоров'я.

Запропонований алгоритм для порівняння зображень з використанням блочного методу обчислення хеш-коду зображення дозволяє відслідковувати прогрес лікування та визначати зміни на томографічних знімках.

Метою роботи є покращення порівняння медичних томографічних зображень з використанням блочного методу обчислення хеш-коду зображень.

Об'єктом дослідження є обробка зображень.

Предметом дослідження є методи порівняння томографічних зображень.

Методи дослідження: методи порівняння зображень, порівняння алгоритмів по критеріям точності, швидкодії, надійності та доцільності використання для порівняння томографічних зображень.

Наукова новизна полягає у модифікації алгоритму Average Hash та застосування блочної обробки зображень як перцептивної хеш-функції для обчислення хеш-коду зображень, що дозволяє порівнювати томографічні знімки, вміст яких може бути оберненим на кут повороту до 7°.

Практична цінність полягає у можливості використання розробленого алгоритму для порівняння зображень, що можуть бути модифіковані шляхом їх обертання на кут до 7°, та доцільності використання такого алгоритму для порівняння томографічних знімків. Алгоритм не є ресурсозатратним та є

швидким у плані часу виконання, тому може бути використаний у високонавантажених системах та системах з обмеженими ресурсами.

Публікації.

- «Алгоритм порівняння зображень з використанням дискретного косинусного перетворення»;
- «Використання дискретного косинусного перетворення як хеш-функції для обчислення хеш-коду зображення».

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів, висновків та додатків.

У вступі представлена загальна характеристика роботи, запропоноване використання методів обробки зображень для медичних зображень, обґрунтована актуальність роботи.

У першому розділі описуються та порівнюються між собою існуючі алгоритми порівняння зображень.

У другому розділі наведені визначення, особливості, властивості та характеристики перцептивних хеш-функцій, хеш-кодів та методів їх порівняння хеш-кодів.

У третьому розділі наведені характеристики, які повинен мати перцептивний хеш-алгоритм для обробки томографічних знімків. Запропонований новий, блочний алгоритм обчислення хеш-коду зображень та перелічені його властивості.

У четвертому розділі наведені результати тестів та досліджень розробленого алгоритму на тестових наборах зображень та їх модифікаціях, а також порівняння розробленого алгоритму з іншими перцептивними хеш-алгоритмами.

У висновках стисло наводяться результати розробки та досліджень.

Ключові слова алгоритм, обробка зображень, порівняння зображень, хеш-код, перцептивна хеш-функція.

ABSTRACT

Topic relevance

Image processing and computer vision are key disciplines of image work and are used to develop real-time applications and systems to solve analytical, research and scientific problems. Medical imaging is an important part of medical research and health care.

The proposed algorithm for comparing images using the block-based method of calculating the hash code of the image allows to track the progress of treatment and determine changes in tomographic images

The object of study is image processing.

The subject of study are image comparison methods.

The goal of this work is an improvement in the comparison of medical tomographic images using the block-based method of calculating hash codes of images.

Study methods: methods of image comparison, comparison of algorithms on criteria of accuracy, speed, reliability and expediency of use for comparison of tomographic images.

The scientific novelty is the modification of the Average Hash algorithm and the use of the block processing of the image is as a perceptual hash function for calculating the hash code of the image, so that it allows to compare the tomographic images, which can be rotated up to 7°.

The practical value is the possibility of using the developed algorithm to compare images that can be modified by rotating them at an angle of up to 7 ° and the feasibility of using such an algorithm to compare tomographic images. The algorithm is not resource intensive and is fast in terms of execution time, so it can be used in high-load systems and systems with limited resources.

Publications.

- “Algorithm for comparing images using discrete cosine transform”;

- “The usage of a discrete cosine transform as a hash function to calculate the hash code of an image".

Structure and scope of work. The master's dissertation consists of an introduction, four chapters, conclusions and appendices.

The introduction presents the general characteristics of the work, the proposed use of image processing methods for medical images, substantiates the relevance of the work.

The first chapter describes and compares existing image comparison algorithms.

The second chapter presents definitions, features, properties and characteristics of perceptual hash functions, hash codes and methods of their comparison of hash codes.

The third chapter presents the characteristics that a perceptual hash algorithm must have for processing tomographic images. A new, block algorithm for calculating the hash code of images and its properties are proposed.

The fourth chapter presents the results of tests and research of the developed algorithm on test sets of images and their modifications, as well as a comparison of the developed algorithm with other perceptual hash algorithms.

The conclusion summarizes the results of development and research.

Keywords algorithm, image processing, image comparison, hash code, perceptual hash function.

ЗМІСТ

Перелік скорочень, умовних познак, одиниць і термінів..... 3
Вступ..... 4
1. Аналіз методів порівняння зображень..... 6
1.1 Порівняння зображень за допомогою хеш-коду..... 6
1.1.1 Алгоритм Average Hash..... 7
1.1.2 Алгоритм Difference Hash..... 9
1.1.3 Алгоритм pHash..... 10
1.1.4 Алгоритм Radial Variance Based Hash..... 13
1.1.5 Алгоритм Marr-Hildreth Operator Based Hash..... 15
1.2 Порівняння зображень за допомогою виявлення ознак та ключових точок..... 16
1.2.1 Scale-Invariant Feature Transform (SIFT)..... 18
1.2.2 Speeded-Up Robust Features (SURF)..... 21
1.2.3 Features from Accelerated Segment Test (FAST)..... 25
Висновки до розділу..... 30
2. Перцептивні хеш-функції та їх використання..... 32
2.1 Поняття медіа-об’єкту та операції над медіа-об’єктами..... 32
2.2 Визначення хеш-функції та її властивості..... 33
2.3 Використання перцептивних хеш-функцій..... 37
2.4 Функції порівняння хеш-кодів..... 41
Висновки до розділу..... 44
3. Розробка алгоритму порівняння томографічних знімків..... 45
3.1 Підстава для розробки..... 45
3.2 Алгоритм Block Truncation Coding та його особливості..... 47
3.3 Розробка блочного алгоритму обчислення хеш-коду зображення..... 52
3.4 Зменшення розміру зображення та його інтерполяція..... 56
3.5 Методи зведення зображення до градацій сірого..... 59
Висновки до розділу..... 61
4. Тестування та дослідження запропонованого алгоритму..... 62
4.1 Вхідні дані та підходи до тестування алгоритму..... 62
4.2 Час виконання..... 63
4.3 Розподіл значень немодифікованих зображень..... 65
4.4 Зміна розміру зображень..... 66
4.5 JPEG стиснення..... 67

4.6 Обертання зображень 68
4.7 Томографічні знімки 69
Висновки до розділу 74
Висновки 75
Список використаної літератури 77

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

Відстань Геммінга – число позицій, у яких відповідні цифри двох двійкових слів однакової довжини різні

ДКП – дискретне косинусне перетворення

Хеш (хеш-код, хеш-сума) – бітовий рядок визначеної довжини, отриманий за допомогою хеш-функції

Хеш-функція – функція, що виконує перетворення масиву вхідних даних довільної довжини в бітовий рядок визначеної довжини.

BER – Bit Error Rate

BTC – Block Truncation Coding

DXTC – DirectX Texture Compression

ID3 – Identity an MP3

FAST – Features from Accelerated Segment Test

JPEG –Joint Photographic Experts Group

PCC – Peak of Cross Correlation

RGB – Red, Grean, Blue

SIFT – Scale-Invariant Feature Transform

SURF – Speeded-Up Robust Features

ВСТУП

Медицина є невіддільною частиною людства та людської діяльності. Вивчення процесів в організмі людини, різноманітних захворювань та патологій, а також їх лікування є основною метою медицини. Це дозволяє краще розуміти людський організм та підтримувати його біологічне функціонування. Це важливо безпосередньо для повсякденного життя людей, а також для функціонування державних та підприємчих процесів.

Існує ряд факторів, які впливають на функціонування охорони здоров'я. Це можуть бути біологічні особливості кожної людини, кваліфікованість спеціалістів, а також наявність засобів та обладнання для лікування. Сьогодні актуальним є використання програмного забезпечення, яке дозволяє:

- обробляти дані різного об'єму та формату;
- подавати результати, аналітику та прогнози;
- економити час та сили спеціалістів та пацієнтів;
- зводити людські фактори до мінімуму.

Якість спеціалізованого програмного забезпечення повинна бути високою, адже від цього залежать життя та здоров'я пацієнтів. Саме тому сфера охорони здоров'я є дуже важливою, відповідальною та фінансово дорогою в обслуговуванні. Методи вирішення повсякденних медичних задач є і будуть актуальними.

Одним із способів вирішення широкого спектра задач є обробка зображень. Обробка зображень – будь-яка форма обробки інформації, для якої вхідні дані представлені зображенням. Обробка зображень може використовуватися як для отримання зображень на виході, так і для отримання іншої інформації.

В медицині широко використовується поняття «Медичне зображення». Це є структурно-функціональний образ органів людини для їх діагностики та вивчення. Одним із типів медичного зображення є томографія, що дозволяє отримати зображення внутрішньої структури організму.

Обробка зображень є широким поняттям та дозволяє вирішувати широкий спектр задач, які можна використати для обробки медичних знімків. Через те, що останнє може мати конкретну мету – необхідно адаптувати загальні методи обробки зображень під вирішення конкретних медичних задач.

Однією з задач по обробці зображень є порівняння декількох або великої кількості зображень. Алгоритми порівняння зображень використовують різні способи ідентифікації та представлення зображень. Результатом таких алгоритмів є кількісна величина, на основі якої робляться прогнози про схожість зображень.

Такі методи є корисними при слідкуванні за прогресом лікування пацієнта. Кожен пацієнт може мати свою історію хвороб, у якій містяться медичних зображеннях. Порівняння таких знімків дозволяє визначити ключові етапи лікування, вказати, на які зображення потрібно звернути увагу в першу чергу, а також знайти схожі знімки інших пацієнтів у базі даних для аналізу.

1. АНАЛІЗ МЕТОДІВ ПОРЯВНЯННЯ ЗОБРАЖЕНЬ

1.1 Порівняння зображень за допомогою хеш-коду

Поняття хеш-коду широко використовується в криптографічних алгоритмах [1]. В криптографії хеш-код являє собою випадкове число (бітовий рядок) фіксованої довжини. Вхідні дані, з яких генерується хеш-код, являють собою джерело випадкових чисел. Тобто, від даних залежить значення хеш-коду, та найменша зміна даних повинна давати інший хеш-код. Якщо хеш-коди однакові – то дані також однакові. Якщо хеш-коди різні – то і дані різні.

Існує можливість колізій хеш-кодів. Річ в тому, що хеш-коди обмежені по своїй довжині, тому, існує мінімальна ймовірність, що для різних даних буде згенеровано однаковий хеш-код.

Для обчислення хеш-коду використовується спеціальна хеш-функція, яка на основі даних вираховує бітовий рядок хеш-коду. Хеш-функція повинна враховувати всі дані, які ідентифікують деякий об’єкт. Тобто, на основі саме цих даних відбувається генерування хеш-коду та порівняння його з іншими хеш-кодами за необхідності.

Хеш-функції використовуються для визначення і числового представлення різних форм мультимедійних даних [2], для виявлення випадків порушення авторських прав в Інтернеті, а також у цифровій криміналістиці завдяки можливості встановлення кореляції між хеш-кодами для знаходження подібних даних. Наприклад, Вікіпедія може мати базу даних текстових хеш-кодів популярних Інтернет-книг або статей, на які автори мають авторські права, в будь-який час, коли користувач Вікіпедії завантажує Інтернет-книгу чи статтю, що має авторські права, хеш-коди будуть майже точно такими ж та матеріал може бути позначеним як плагіат. Цю саму систему позначення

можна використовувати для будь-якого мультимедійного або текстового файлу, а також для виявлення подібного вмісту медіафайлів, для виявлення копіювання відеофайлів та визначення зловмисних маніпуляцій над відеоконтентом. Хеш-коди можуть бути обчисленні одноразово, занесені в базу даних та використовуватися для швидкого порівняння медіа-файлів.

Також хеш-коди використовуються для ідентифікації зображень та для перевірки їх цілісності, порівняння з іншими зображеннями та пошуку схожих. У наступних підрозділах розглянуті алгоритми обчислення хеш-коду зображень.

1.1.1 Алгоритм Average Hash

Розглянемо базовий алгоритм обчислення хеш-коду для зображення під назвою Average Hash [3], та визначимо його переваги та недоліки. Алгоритм складається з наступних кроків:

1) Зменшити розмір зображення. Це дозволяє швидко та ефективно позбавитися від високих частот, тобто, від деталізації зображення, залишивши лише «основу» зображення. Зазвичай зменшують до розміру 8x8, 32x32. Для прикладу, розглянемо зменшення до розміру 8x8;
2) Видалити кольорову складову. Зменшене зображення переводиться в градації сірого. Тобто, позбавляємося від значень красного, зеленого та синього кольорів (RGB) та зводимо зображення до 64 (8x8) значень відтінків сірого;
3) Обчислити середнє значення всіх 64 кольорів;

4) Побудувати бітовий рядок довжиною 64, в якому кожен біт приймає значення «0», якщо значення кольору менше за середнє, та значення «1», якщо більше за середнє;
5) Отриманий хеш-код перевести в 64-бітне число.

Зменшення розміру зображення дозволяє порівнювати зображення, навіть якщо одному із них змінили розмір або співвідношення сторін.

Якщо обчислити хеш-код для одного і того ж зображення декілька раз – отримаємо однакові значення хеш-кодів, які можуть бути порівнянні. Але якщо над зображенням проводили мінімальні маніпуляції - то необхідно визначати різницю між хеш-кодами отриманих зображень. Одним зі способів порівняння хеш-кодів є відстань Геммінга [4]. Відстань Геммінга – число позицій, у яких відповідні цифри двох двійкових слів однакової довжини різні. Відстань Геммінга зі значенням 0 гарантує, що значення хеш-кодів однакове, та, відповідно, зображення однакові.

Відповідно до поставленої задачі можна визначити граничну відстань Геммінга. Якщо значення відстані Геммінга нижче граничного, то можна стверджувати, що одне зображення «схоже» на інше зображення та навпаки.

Перевагою даного алгоритму є простота та швидкість виконання. Даний алгоритм видає кількісну, вимірювальну різницю між зображенням, яку можна порівнювати та аналізувати. Алгоритм інваріантний до:

- стиснення зображення (наприклад, алгоритмом JPEG);
- розмиття Гауса;
- електронних фільтрів;
- зміни розміру зображення;
- зміни співвідношення сторін зображення.

Однак, навіть при незначних модифікаціях зображення алгоритм не спроможний дати коректний результат при зіставленні оригінального

зображення з модифікованим. Тобто, алгоритм не є інваріантним до наступних модифікацій:

- модифікації зображення шляхом нанесення надписів, замальовування частин зображення тощо (тобто, модифікації вмісту зображення) (при модифікації більше ніж 5% площі);
- обрізання зображення (при модифікації більше ніж 5% площі);
- обертання зображення (при обертанні на кут, більший 2°);
- масштабування зображення[3].

Тобто, цей алгоритм доцільно використовувати лише для певних задач, для яких перечислені недоліки не є критичними. В такому випадку, завдяки своїй простої реалізації та швидкодії, можливо досягнути необхідної мети максимально швидко та ефективно.

1.1.2 Алгоритм Difference Hash

Інший варіант порівняння зображень за допомогою хеш-коду – використання різниці значень між сусідніми пікселями зображення для генерування хеш-коду [3]. Це визначає відносний напрямок градієнту.

Розглянемо базовий алгоритм обчислення хеш-коду для зображення таким методом та визначимо його переваги та недоліки.

1) Зменшити розмір зображення. Це дозволяє швидко та ефективно позбавитися від високих частот, тобто, від деталізації зображення, залишивши лише «основу» зображення. Для цього алгоритму зображення зменшується до розміру 9x8.

2) Видалити кольорову складову. Зменшене зображення переводиться у градації сірого. Тобто, позбавляємося від значень красного, зеленого та синього кольорів (RGB) та зводимо зображення до 72 (9x8) значень відтінків сірого;

3) Обчислити різницю. Алгоритм використовує визначення різниці між сусідніми пікселями. В такому випадку, 9 пікселів на рядок дають 8 різниць між сусідніми пікселями. 8 рядків з восьми різниць дають 64 біти;

4) Побудувати бітовий рядок довжиною 64, в якому кожен біт приймає значення «0», якщо значення лівого пікселя більше значення правого, та значення «1», якщо більше за середнє. Тобто, порівнюються значення
яскравості сусідніх пікселів;

Як і у випадку з попереднім алгоритмом, хеш-коди можуть бути порівняні між собою за допомогою відстані Геммінга[4], що дозволить визначити кількісну різницю між хеш-кодами, та, відповідно, зображеннями.

Алгоритм має недоліки та переваги алгоритму Average Hash за виключенням масштабування зображень (у межах коефіцієнтів масштабування від 0,5 до 2).

Основним недоліком на ряду з алгоритмом Average Hash є те, що навіть незначні модифікації зображення, такі як додавання надписів, замалювання частин зображення, обрізання зображення або його нахил ведуть до неспроможності алгоритму порівняти зображення між собою. Також даний алгоритм не є інваріантним до обрізання та обертання зображень, що також є вагомим недоліком для вирішення певного спектру задач.

1.1.3 Алгоритм pHash

Алгоритм pHash [5] схожий на алгоритм Average Hash. Однак, даний алгоритм використовує дискретне косинусне перетворення другого типу для визначення низьких частот зображення, що дозволяє ефективніше визначити ключову складову зображення та забезпечити кращі результати при порівнянні зображень з модифікованим вмістом.

Алгоритм зводиться до наступних кроків:

1) Зменшити зображення до розміру 32x32. Експериментально встановлено, що розмір 32x32 є оптимальним. Така розмірність матриці дозволяє з необхідною точністю розподілити значення частот, що є результатом дискретного косинусного перетворення. Можливе зменшення зображення до інших розмірів (64x64, 128x128). Більший розмір зображення дозволяє збільшити довжину результуючого хеш-коду у бітах, але ускладньює обчислення ДКП у плані швидкодії. Зменшення розміру зображення необхідне для спрощення обчислення ДКП та його прискорення.
2) Перевести зображення у градації сірого. В результаті, хеш-код зменшиться в 3 рази. З 64 значень красного, 64 зеленого та 64 синього до 64 значень відтінків сірого;
3) Використати дискретне косинусне перетворення для отриманого зображення. Згідно з пунктами 2-3, зображення представлене у вигляді матриці 32x32. Кожен елемент матриці – значення кольору. У результаті перетворення отримаємо набір частот і векторів.

Нехай x[m], m = 0, …, N – 1 визначають N-точкову послідовність реальних сигналів. Тоді формула (1) виражає дискретне косинусне перетворення типу II (ДКП-2) [6]:

$$X[n] = \sqrt{\frac{2}{\mathrm{N}}} * \sum_{m=0}^{N-1} \left(x[m] * cos\left(\frac{(2m+1) * n\pi}{2N} \right) \right) \qquad (1.1)$$

де n = 0, …, N – 1.

5) Скоротити результат дискретного косинусного перетворення. Результат, поданий у вигляді 32x32, необхідно скоротити результат до підматриці 8x8, яка береться із верхнього лівого кута основного результату, тобто [0, 7][0, 7]. Скорочення матриці дозволяє позбавитися високих частот, тобто даних, що несуть в собі деталізовану складову зображення. Приклад результативної підматриці наведено на рис. 1;
6) Обчислити середнє значення значень всіх елементів отриманої підматриці. Важливий момент при цьому – не враховувати елемент матриці на позиції [0, 0], тому що цей елемент несе в собі найбільш загальну інформацію (наприклад, однакові кольори). Ця інформація не є ключовою та не визначає зображення.

   Примітка. На рис. 1.1 можна помітити, що цей елемент не несе суттєвої інформації;

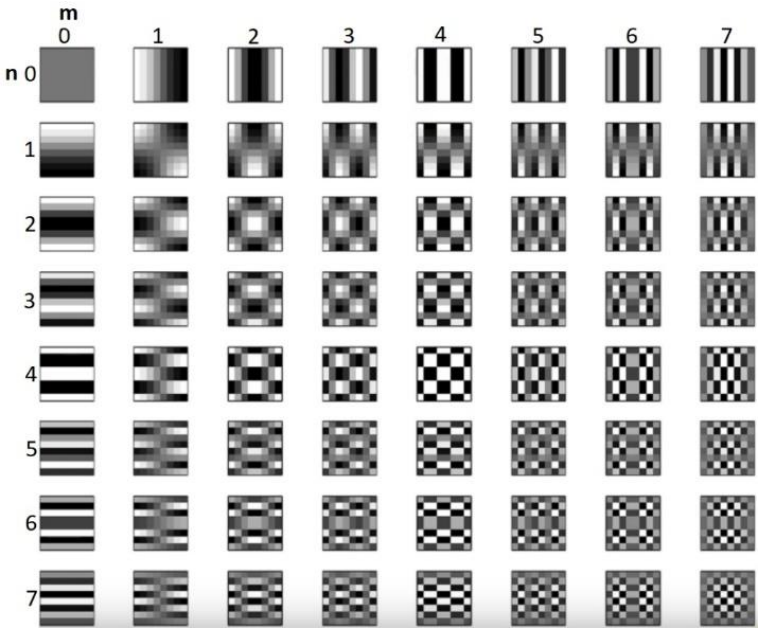

Рисунок 1.1 - Підматриця DCT частот 8x8

7) Розгорнути підматрицю у двійковий вектор, де кожен елемент приймає значення «0», якщо значення відповідного елементу підматриці менше ніж середнє значення, та значення «1», якщо більше середнього значення. Вектор, що утворився в результаті, є хеш-кодом даного зображення.

Алгоритм має переваги та недоліки попередніх алгоритмів, але алгоритм pHash показує кращі результати при порівнянні оригінального зображення з модифікованим шляхом зміну вмісту зображення (видалення частини площі зображення, додавання надписів, клякс тощо) до 15% площі. Це є результатом переваг використання ДКП. Також, даний алгоритм показує кращий статистичний розподіл значень на великій вибіркі зображень.

1.1.4 Алгоритм Radial Variance Based Hash

Ідея алгоритму полягає в побудові променевого вектора дисперсії (ПВД) на основі перетворення Радону [7]. Потім до ПВД застосовується ДКП та обчислюється хеш-код. Перетворення Радону – це інтегральне перетворення функції багатьох змінних уздовж прямої. Перетворення стійке до обробки зображень за допомогою різних маніпуляцій (наприклад, стиснення). Кроки алгоритму:

1) Прибрати колір, щоб позбавитися від непотрібних високих частот. Перевести зображення у градації сірого;
2) Розмити зображення за допомогою розмиття Гауса для приглушення шумів. Рівняння Гауса для двовимірного випадку складається з двох одномірних рівнянь Гауса, по одній для кожної осі виміру (1.2):

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1.2}$$

де x — це відстань від початку координат в осі абсцис, y — це відстань від початку координат у осі ординат, а σ являє собою середньоквадратичне відхилення розподілу Гауса.

3) Застосувати гамма-корекцію для усунення бляклості зображення. Гамма-корекція - коригування яскравості цифрового зображення або відеопотоку. Зазвичай, використовується степенева функція у вигляді (1.3):

$$V_{out} = AV_{in}^{\gamma} \tag{1.3}$$

де A – коефіцієнт, V_{in} – вхідні значення, V_{out} – вихідні значення, що є дійсними числами. Якщо A = 1, то вхідні та вихідні значення знаходяться в межах від 0 до 1. Якщо $\gamma = 1$, то характеристика передачі полутонів лінійна, а також перепади освітленості об'єкту відображуються однаково.

4) Побудувати променевий вектор дисперсії. Нехай I(x, y) позначає яскравість пікселя, а G(α) – потужність множини. Тоді променевий вектор дисперсії R(α), де α = 0, 1, ..., 179 (1.4):

$$R(\alpha) = \frac{\sum_{(x,y)\in G(\alpha)} I^2(x,y)}{G(\alpha)} - \left(\frac{\sum_{(x,y)\in G(\alpha)} I(x,y)}{G(\alpha)}\right)^2 \qquad (1.4)$$

5) Застосувати ДКП до вектора дисперсії (1.1);
6) Побудувати хеш-код. Для порівняння хеш-кодів цього типу використовується пошук піка взаємно кореляційної функції.

У порівнянні з алгоритмо pHash, розглянутий алгоритм показує гірший статистичний розподіл значень на великій вибіркі зображень, більш чутливий до зміни розміру зображення та його обертання. Але такий алгоритм показує кращі результати при модифікації зображення шляхом стиснення JPEG.

1.1.5 Алгоритм Marr-Hildreth Operator Based Hash

Оператор Марра–Хілдрета дає змогу визначати краї на зображенні [7][8]. Край зображення – це область зображення, що відокремлює сусідні

частини зображення, які мають порівняно відмінні характеристики відповідно до деяких особливостей. Цими особливостями можуть бути колір або текстура, але найчастіше це сіра градація кольору зображення (яскравість). Результатом визначення меж є карта кордонів. Карта кордонів описує класифікацію меж для кожного пікселя зображення. Якщо межі визначати як різку зміну яскравості, то для їх знаходження використовують похідні або градієнт. Кроки алгоритму:

1) Звести зображення до градацій сірого;
2) Зменшити зображення. У запропонованому автором варіанті зображення зменшується до розміру 128×128;
3) Розмити зображення за допомогою розмиття Гауса (1.2);
4) Побудувати оператор Марра–Хілдрета (1.5):

$$h_c(x,y) = \nabla^2 g_c(x,y) = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} * e^{\frac{x^2+y^2}{2\sigma^2}} \quad (1.5)$$

5) Застосувати дискретну згортку до LoG (Laplacian Of Gaussian) та до зображення. Отримаємо зображення, на якому чітко видно стрибки яскравості. Нехай x, y, z – піксельна ширина, довжина та глибина зображення. Результат дискретної згортки зображення I з маскою M визначається як (1.6):

$$R(x,y,z)\sum_{i,j,k} I(x-i, y-j, z-k) * M(i,j,k) \quad (1.6)$$

6) Перетворити зображення в гістограму. Зображення розбивається на маленькі блоки (5×5), в яких підсумовуються значення яскравостей.

7) Побудувати хеш із гістограми. Гістограма розбивається на блоки 3×3. Для цих блоків розраховується середнє значення яскравості і використовується метод побудови ланцюжка бітів. Виходить бінарний хеш розміром 64 байти.

Порівняння двох хеш-кодів займає небагато часу, не зважаючи на те, що їх довжина є відносно великою, оскільки використовується функція нормованої відстані Геммінга.

Даний алгоритм дає схожі результати до алгоритму pHash. Однак, алгоритм показує кращий статистичний розподіл значень на великій вибіркі зображень (приблизно на 15-20%). Однак, у порівнянні з алгоритмом pHash, даний алгоритм більш чутливий до JPEG стиснення та обертання.

1.2 Порівняння зображень за допомогою виявлення ознак та ключових точок

Методи для порівняння зображень, основані на порівнянні зображень в цілому та розглянуті у попередніх підрозділах. Такі методи представляють зміст зображення за допомогою визначених функцій, результат яких може порівнюватися. Такі методи доцільно використовувати в обмежених випадках, розуміючи всі їх переваги, недоліки та обмеження. Поява нових об'єктів на зображенні, перекриття окремих об'єктів іншими, шуми, зміна положення об'єкта на зображенні, положення камер у тримірному просторі приводять до неможливості використання таких алгоритмів для більш комплексних випадків.

Тому треба або вибирати точки, що роблять внесок в характеристику, або виділяти деякі особливі (ключові) точки і порівнювати їх. На цьому базується ідея зіставлення зображень за ключовимими точкам. Можна сказати, що замінюється зображення деякої моделі – набором його ключових точок. Відразу відзначимо, що ключовою буде називатися така точка зображеного об'єкта, яка з великою часткою ймовірності буде знайдена на іншому зображенні цього ж об'єкта. Детектором будемо називати алгоритм виявлення ключових на зображенні. Детектор повинен забезпечувати інваріантність знаходження одних і тих же ключових щодо перетворень зображень.

Важливо зазначити те, що в україномовних джерелах використовуються два терміни – «ключові точки» та «особливі точки». Також, в деяких випадках зустрічається термін «особливі точки». Англійською цей термін має назву «keypoints» [10]. Тому «найближчим» терміном є термін «ключові точки», який використовується в даній роботі.

Єдине, що залишається незрозумілим – яким чином визначати, яка ключова точка одного зображення відповідає ключовій точці іншого зображення. Адже після застосування детектора можна визначити тільки координати ключових точок, а вони на кожному зображенні різні. Тут в справу і вступають дескриптори. Дескриптор - ідентифікатор ключової точки, що виділяє її з іншої маси ключових точок. У свою чергу, дескриптори повинні забезпечувати інваріантність знаходження відповідності між ключовими точками щодо перетворень зображень.

У результаті маємо наступну схему розв’язання задачі зіставлення зображень:

1) На зображеннях визначити ключові точки та їх дескриптори.
2) За збігом дескрипторів визначаються відповідні один одному ключові точки.

3) На основі набору ключових точок, що зівпали, будується модель перетворення зображень, за допомогою якого з одного зображення можна отримати інше.

У наступних підрозділах розглянуті найбільш відомі та ефективні алгоритми виявлення клювочих точок, такі як SIFT, SURF та FAST.

1.2.1 Scale-Invariant Feature Transform (SIFT)

Масштабно-інваріантне перетворення ознак (SIFT) - це алгоритм виявлення ознак у сфері комп'ютерного зору для виявлення та опису локальних особливостей зображень. Алгоритм був опублікований Девідом Лоу [9]. Автори в [11,12] дослідили SIFT для порівняння зображень та можливі модифікації алгоритму. Програми з використанням даного алгоритму включають розпізнавання об’єктів, роботизовану картографію та навігацію, зшивання зображень, 3D-моделювання, розпізнавання жестів, відстеження відео, індивідуальну ідентифікацію тощо.

Ключові точки SIFT об’єктів спочатку визначають з набору опорних (контрольних) зображень і зберігають в базі даних. Об’єкт розпізнається в новому зображенні шляхом індивідуального порівняння кожної ознаки з нового зображення до цієї бази даних та пошуку ознак, що відповідають кандидатам, на основі евклідової відстані їх векторів ознак. З повного набору збігів виділяються ключові точки, які узгоджуються щодо об’єкта та його розташування, масштабу та орієнтації в новому зображенні для виявлення збігів. Визначення послідовних блоків виконується швидко за допомогою ефективної реалізації хеш-таблиці узагальненого перетворення Хафа. Кожен

блок із трьох або більше ознак, що узгоджують об'єкт та його позицію, підлягає подальшій детальній верифікації моделі, а згодом відкидають незгідні блоки. Нарешті, обчислюється ймовірність того, що певний набір ознак вказує на наявність об’єкта, враховуючи точність збігу та кількість ймовірних помилкових збігів. Збіги об’єктів, які пройшли всі ці тести, можна з високою ймовірністю визнати правильними.

Для будь-якого об'єкта в зображенні можуть бути взяті ключові точки для забезпечення «опису ознаки» об'єкта. Це опис, отриманий з тренувального зображення, може бути потім використаний для впізнання об'єкта для визначення місця об'єкта у тестовому зображенні, що містить багато інших об'єктів. Для здійснення надійного розпізнавання важливо, щоб ознаки, взяті з тренувального зображення, могли бути виявлені навіть при змінах в масштабі зображення, зміні шуму та освітлення. Такі точки зазвичай лежать в областях високої контрастності, таких як границі об'єктів.

Інша важлива характеристика цих ознак, що відносні позиції між ними не повинні змінюватися від одного зображення до іншого. Наприклад, якщо тільки чотири кути двері використовувалися як ознаки, вони будуть працювати незалежно від положення дверей. Але якби точки косяка двері також використовувалися, розпізнавання могло б зазнати невдачі, оскільки двері можуть бути відкритими або закритими. Аналогічно, ознаки, розміщені на шарнірно закріплених або гнучких об'єктах, як правило, не працюють, якщо будь-яка зміна у внутрішній геометрії трапляється між двома зображеннями в оброблюваному наборі. Однак, на практиці, SIFT виявляє і використовує багато більше число ознак зображень, що зменшує внесок кожної помилки, викликаної цими локальними змінами в загальній помилці всіх помилок відповідності ознак.

SIFT може надійно виділити об'єкти навіть при наявності шуму і часткового перекриття, оскільки дескриптор SIFT інваріантний до пропорційного масштабування, орієнтації, зміни освітленості і частково інваріантний афінним перетворенням.

Основні кроки алгоритму:

1) Виявлення масштабно-інваріантних ознак. Метод Лоу для генерації ознак зображення перетворює зображення в великий набір векторів ознак, кожен з яких інваріантний щодо (паралельного) перенесення зображення, масштабування і обертання, частково інваріантний до зміни освітлення і стійкий до локальних геометричних спотворень;
2) Зіставлення ознак та індексація. Індексація складається з запам'ятовування SIFT-ключів і ідентифікації відповідних ключових точок нового зображення. Використовується модифікація алгоритму k-мірного дерева, який називається методом пошуку best-bin-first, який може ідентифікувати найближчого сусіда з великою ймовірністю виконуючи лише обмежену кількість обчислень.;
3) Ідентифікація кластера за допомогою перетворення Хафа. Перетворення Хафа використовуються для кластеризації надійної моделі гіпотез для пошуку ключів, які узгоджуються з конкретної позиції моделі. Перетворення Хафа виявляє кластери ознак з узгодженим тлумаченням за допомогою голосування за кожну ознаку для всіх положень об'єкта.
4) Перевірка моделі методом найменших квадратів. Кожен встановлений кластер є предметом процедури перевірки, в якій здійснюється рішення методу найменших квадратів для параметрів афінного перетворення, пов'язаних з моделлю зображення;

5) Виявлення викидів (промахів), що відбувається шляхом перевірки узгодження між ознакою кожного зображення і моделі, заданого рішенням параметрів.

Остаточне рішення щодо прийняття або відкидання моделі гіпотези грунтується на детальній ймовірнісної моделі. Цей метод спочатку обчислює очікуване число помилкових відповідностей моделі положення, заданою розміром моделі, числом ознак всередині області і точністю відповідності. Байєсовий аналіз визначає ймовірність, що об'єкт присутній, ґрунтуючись на дійсне число знайдених відповідностей ознак. Модель приймається, якщо кінцева ймовірність правильної інтерпретації більше 0,98. Метод SIFT розпізнавання об'єкта дає прекрасні результати, за винятком випадків широкого розкиду освітлення та сильно модифікованих зображень.

В таблиці 1 наведені основні задачі, які можуть ставитися перед алгоритмом, та техніки їх рішень та переваги таких рішень.

Таблиця 1 – SIFT задачі, техніки їх вирішення та їх переваги

Задача	Техніка	Переваги
Масштабування/обертання	Визначення орієнтації/різниця Гаусіанів	Інваріантність до масштабу і обертання, точність
Геометричне спотворення	Розмиття площин орієнтації локального зображення	Інваріантність до геометричних спотворень

Індексація і зіставлення	Метод «Best Bin First» та метод «найближчого сусіда»	Швидкість та ефективність
Ідентифікація кластера	Перетворення Хафа	Надійні моделі
Верифікація моделі	Лінійний метод найменших квадратів	Менша ймовірність похибки та менша кількість порівнянь
Прийняття гіпотези	Баєсовий ймовірнісний аналіз	Надійність

1.2.2 Speeded-Up Robust Features (SURF)

Алгоритм SIFT є недостатньо ефективним для вирішення певного роду задач. Виникла необхідність у розробці нового алгоритму визначення ключових точок. У Бей, Х., Тюйтелаарс, Т. та Ван Гоол, Л., опублікували статтю "SURF: Speeded Up Robust Features" [10], яка представила новий алгоритм під назвою SURF для виявлення. Як випливає з назви, цей алгоритм являє собою прискорену версію алгоритму SIFT. Основний інтерес до підходу SURF полягає в його швидкому обчисленні операторів та можливість його використання для роботи додатків в режимі реального часу для відстеження та розпізнавання об'єктів.

В основі SURF лежать SURF детектор, SURF дескриптор та SURF компаратор.

SURF детектор використовується для виявлення blob-подібних структур зображення. Такі структури можуть бути знайдені на кутах об'єктів зображення, а також в зонах, в яких відбиття світла від дзеркальних поверхонь максимальне. Оператор Монжа-Ампера та похідні Гауссові фільтри використовують для ідентифікації ключових точок. Ключові точки можуть бути знайдені завдяки згорткі використовуючи детермінант Гауссової матриці (DoH), що містить різні Гауссові похідні різного порядку. Результат ділиться на Гаусову різницю σ для нормалізації результату (1.7).

$$DoH(x, y, \sigma) = \frac{G_{xx}(x, y, \sigma) * G_{yy}(x, y, \sigma) - G_{xy}(x, y, \sigma)^2}{\sigma^2} \quad (1.7)$$

де $G_{ij}(x, y, \sigma) = \frac{\partial N(o,\sigma)^2}{\partial i * \partial j} * image(x, y)$

Локальний максимум цього фільтру має місце в регіонах, де обидва значення G_{xx} та G_{yy} додатні, а G_{xy} – негативне. Екстремум знаходиться в регіонах зображення з максимальною інтенсивністю різниць градієнту в декількох напрямках.

Інша причина, чому використовують фільтр Гауса для ідентифікації точок інтересу – позбавлення від зашумлених даних завдяки розмиттю зображення.

Автори SURF та багато інших дослідників виявили, що кількість результатів у масштабному просторі експоненційно зменшується із збільшенням розміру масштабу. Одне з можливих пояснень полягає в тому, що занадто сильне розмиття Гауса може усереднити майже всю корисну інформацію на зображеннях. Отже, лінійний пошук у масштабному просторі може бути безкорисним обчислювально. Як альтернативу, SURF вводить

поняття масштабних октав: кожна октава лінійно відбирає невелику область масштабного простору, але цей розмір області пропорційний використаним величинам масштабу. Кожна сусідня октава подвоює розмір області та приріст вибірки, використаний у попередній, для зменшення обсягу пошуку, необхідного у більших масштабах. Єдиним негативним побічним ефектом використання октав є те, що результати, знайдені у більших масштабах, потенційно можуть мати більше помилок через більші прирости, використані у відповідних октавах. Для компенсації детектор SURF інтерполює координати будь-яких локальних максимумів, знайдених в діапазоні субпікселя.

Нарешті, автори SURF пропонують провести різницю між яскравими пікселями, що знаходяться на темному фоні, від темних пікселів, що знаходяться на світлому фоні. Ця властивість може бути представлена оператором Лапласа, як показано нижче (1.8).

$$sgn\{G_{xx}(x,y,\sigma) + G_{yy}(x,y,\sigma)\} = \begin{cases} +1 => \text{світлий піксель на темному фону} \\ -1 => \text{темний піксель на світлому фоні} \end{cases} \quad (1.8)$$

Таким чином, алгоритм детектора SURF може бути описаний наступними кроками:

1) Визначити значення масштабу-простору, згортаючи зображення використовуючи DoH-фільтри з різними σ ;

2) Знайти локальні максимуми для сусідніх пікселів та сусідніх масштабів в різних октавах;

3) Інтерполювати розташування кожного знайденого локального максимуму;

4) Для кожної визначної точки результатом є x, y (координати) σ (різниця Гаусса), величина DoH (детермінант матриці Гаусса) та значення Лапласіана.

Для опису кожної ключової точки SURF узагальнює інформацію про пікселі в рамках локальної зони навколо пікселя. Першим кроком є визначення орієнтації для кожної ознаки шляхом згортання пікселів використовуючи горизонтальні і вертикальні фільтри вейвлета Хаара, наведені на рисунку 1.2. Дані фільтри можна розглядати як блокові методи для обчислення напрямлених похідних інтенсивності зображення. Використовуючи зміни інтенсивності для характеристики орієнтації, цей дескриптор може описувати об’єкти однаково, незалежно від конкретної орієнтації об’єктів або камери. Ця властивість інваріантності обертання дозволяє функціям SURF точно ідентифікувати об’єкти на зображеннях, зроблених з різних перспектив.

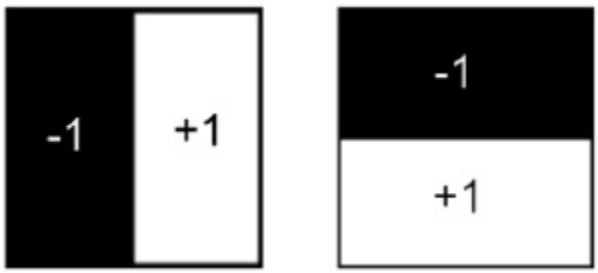

Рисунок 1.2 – Горизонтальні та вертикальні вейвлетні фільтри Хаара

Інформація про градієнт інтенсивності може також надійно характеризувати дані піксельні області. Аналізуючи нормалізовані градієнтні значення, ознаки зображень, зроблених у темній кімнаті у порівнянні зі світлою кімнатою, та інші зображення зроблені з використанням різних налаштувань експозиції - усі такі зображення матимуть однакові значення дескриптора. Тому, використовуючи результат вейвлетів Хаара для

формування вектора, що представляє кожну ознаку та її зону, SURF успадковує дві бажані властивості: інваріантність освітлення та інваріантність контрасту.

Оскільки зони можна розбити на менші зони довільними методами, оскільки окремі результати вейвлетів Хаара можна узагальнити по різному - запропоновано кілька варіантів SURF з використанням різних комбінацій налаштувань. Крім того, в таких додатках, як мобільний роботизований зір, зображення знімаються за допомогою статично розташованих камер, тому всі ці фотографії орієнтовані в однакових напрямках. Для зменшення обчислювальних затрат пропонується варіант алгоритму під назвою U-SURF (де U означає «up-right»), відмовляється від кроку орієнтації та обчислює горизонтальні та вертикальні відповіді вейвлета Хаара, використовуючи безпосередньо основні осі зображення. Експериментально визначено, що ознаки U-SURF алгоритму інваріантні до орієнтації до 15°.

Наприклад, маємо бібліотеку зображень з міченими об’єктами, де кожне зображення містить один об’єкт у повному огляді без перешкод, на простому темному фоні. Кожен об'єкт представлений декількома зображеннями, які зроблені з дещо різних ракурсів. Підхід до розпізнавання об'єктів полягає в обчисленні бази даних ознак для кожного зображення в бібліотеці. На вхід програмі приходить запитуване зображення, генерується набір ознак цього зображення та ставиться у відповідність іншим наборам ознак у базі даних. Після того, як об’єкт бази даних з найкращим збігом об’єктів на основі певної метрики порівняння знайдений, робиться висновок, що цей об’єкт присутній на запитуваному зображенні.

Даний підхід є схожим на алгоритми ідентифікації на основі зовнішнього вигляду, використовуючи Principal Component Analysis (PCA). Кожен об'єкт може бути представлений чисельно у великому розмірному

просторі ваг основних компонентів. Для визначення об’єктів за допомогою PCA зображення запиту проектується на основі компонентів власного простору кожного об’єкта в базі даних, а відповідний вектор ваги порівнюється з існуючими. Ідентичність об’єкта до найближчого числового значення визначається як збіг розпізнавання.

Використовуючи ознаки, кожне зображення визначається набором векторів ознак, а не єдиним вектором ваги. Оскільки значення окремих дескрипторів об’єктів не пов’язані чисельно, кожен об’єкт відповідає хмарі в просторі об’єктів, що містить об’єкти, що належать до всіх його зображень. Аналогічно, дані запиту також можна розглядати як хмару ознак. Використовується результат обчислення евклідової відстані найближчої сусідньої ознаки для оцінки відстані між хмарами точок.

1.2.3 Features from Accelerated Segment Test (FAST)

FAST - це алгоритм, запропонований для ідентифікації ключових точок на зображенні [14]. Точка інтересу на зображенні - це піксель, який має чітко визначену позицію і може бути однозначно визначеним. Ключові точки інформативні щодо розташування та повинні повторюватися на різних зображень Виявлення ключових точок використовується у програмах для пошуку зображень, розпізнаванні об'єктів, відстеженні об’єктів тощо. Ідея виявлення точок інтересу або виявлення кутів (обидва терміни є взаємозамінними в літературі), не є новою.

Причиною розробка алгоритму FAST була розробка детектора ключових точок для використання в програмах реального часу, таких як SLAM, що

використовується для мобільних роботів, у яких обмежені обчислювальні ресурси.

Алгоритм виявлення ключових точок наступний:

1) Вибрати піксель «p» на зображенні. Припустимо, що інтенсивність цього пікселя складає I(p);
2) Встановити порогову інтенсивність T (наприклад, 20%);
3) Розглянуті коло з 16 пікселів, що оточують піксель p. Таке коло являє собою коло Брезенхема с радіусом 3 (рис. 1.3);
4) Значення інтенсивності N суміжних пікселів з наведених 16 повинні бути більше або менше інтенсивності I(p) на значення T. Якщо умова виконується – піксель p являє собою ключову точку. В першій версії алгоритму N має значення 12;
5) Для підвищення швидкодії алгоритму, спочатку порівнюються інтенсивності пікселів 1, 5, 9 та 13 наведеного кола зі значенням I(p). Виходячи з попереднього пункту, щонайменше 3 пікселі повинні задовольняти критерію порогової інтенсивності T;
6) Якщо інтенсивності щонайменше трьох з чотирьох пікселів з попереднього пункту менше або більше I(p) + T – піксель p не являє собою ключову точку. Якщо хоча б три з чотирьох пікселів задовольняють умові – виконується порівняння всіх 16 пікселів для визначення, чи хоча б 12 з них не задовольняють умові. Якщо задовольняють – піксель p вважають ключову точку;
7) Кроки 1- 7 повторюються для всіх пікселів зображення.

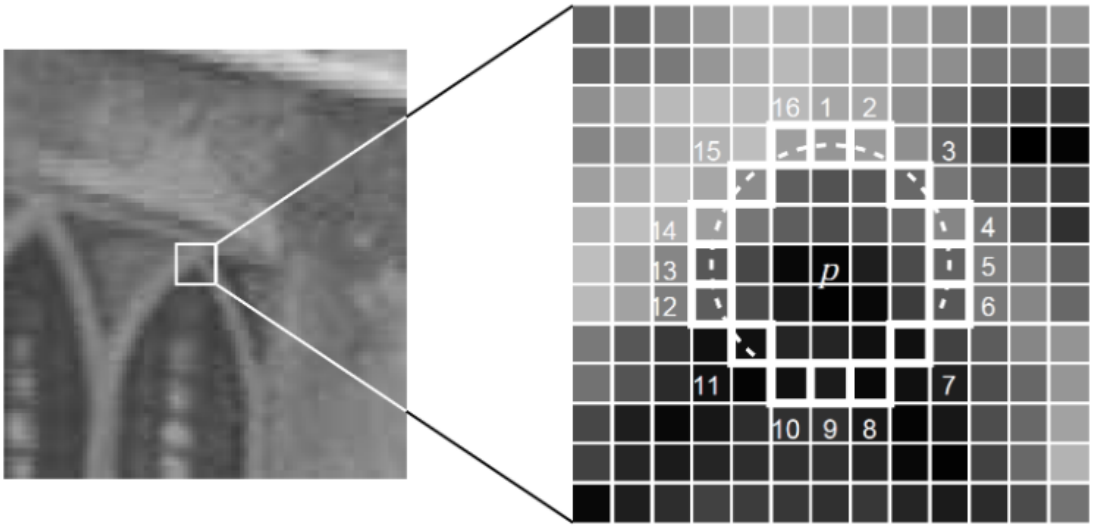

Рисунок 1.3 – Приклад точки інтересу та кола Брезенхема навколо неї

Алгоритм має декілька обмежень. Перш за все, при значенні N < 12 алгоритм працює не завжди коректно та виявляє занадто багато ключових точок. Другим обмеженням є те, що порядок, за яким вибираються пікселі із 16 пікселів для порівняння, визначає швидкодію алгоритму.

Через те, що алгоритм несе собі ціль використовуватися в системах реального часу, для підвищення його швидкодії використовується машинне навчання. Підхід використання машинного навчання з використанням FAST-алгоритму наступний:

1) Визначити набір зображень для тренування нейронної мережі;
2) На кожному зображенні застосувати FAST-алгоритм для визначення ключових точок. Для одного пікселю в момент часу визначаються 16 пікселів кола Брезенхема навколо нього;
3) Для кожного пікселю p зберігаємо значення 16 пікселів кола Брезенхема як вектор (рис. 1.);
4) Попередній крок повторюється для всіх пікселів зображення. Результуючий вектор P містить всі необхідні дані для навчання;

5) Кожне значення у векторі P може мати три стани. Виходячи зі значення,

   піксель кола може бути темніший за p, світліший за p або ідентичний p (1.9);

$$S_{p\to x}\begin{cases} d, I_{p\to x} \le I_p - t \text{ (темніший)} \\ s, I_p - t < I_{p\to x} < I_p + t \text{ (однаковий)} \\ b, I_p + t \le I_{p\to x} \text{ (світліший)} \end{cases}$$

   (1.9)

6) Відповідно до цього, весь вектор P розбивається на 3 вектори P(d), P(s) та P(b);
7) Визначити змінну K(p), що приймає значення «true», якщо p – ключова точка та «false», якщо p не є ключовою точкою;
8) Використати ID3 алгоритм (decision tree classifier) для обробки P(d), P(s) та P(b) векторів;
9) Процес закінчується, коли ентропія P(d), P(s) та P(b) векторів дорівнює нулю;

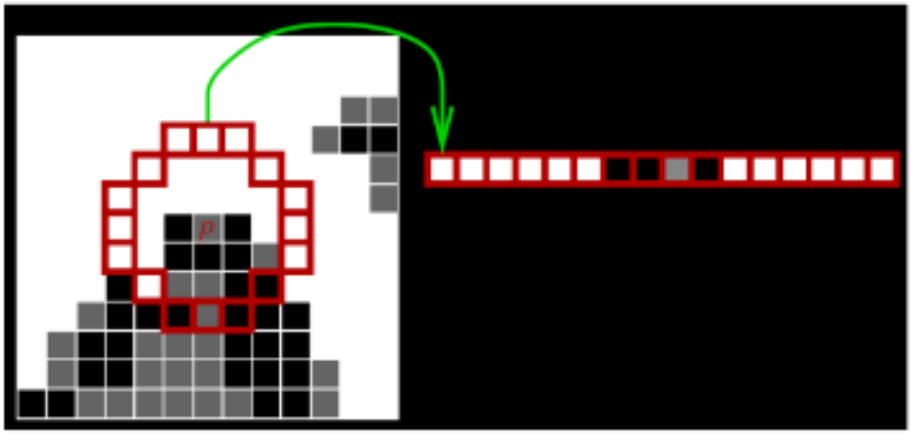

Рисунок 1.4 – Ключова точка та пікселі навколо неї, значення яких записуються у вектор

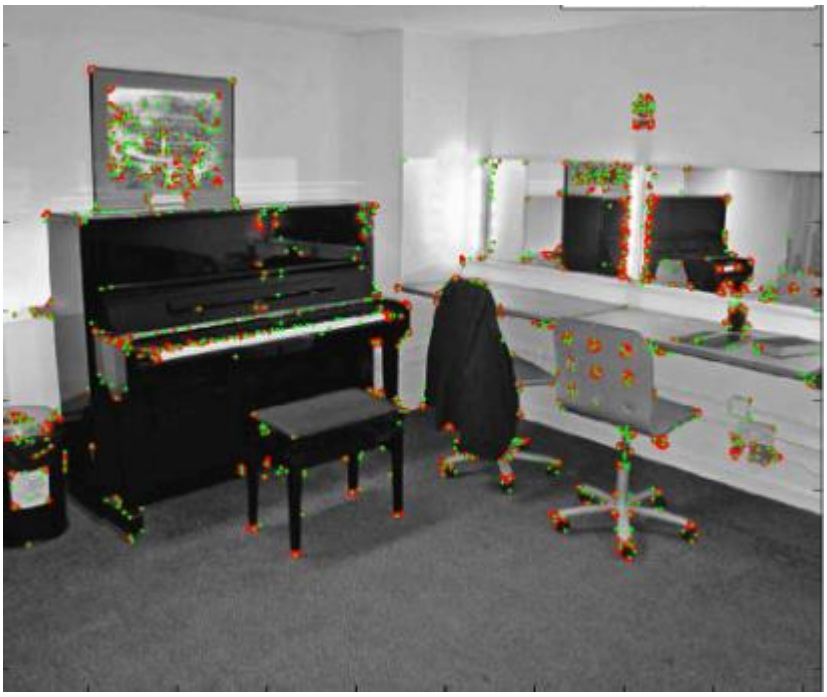

Рисунок 1.5 – Приклад визначених ключових точок за допомогою алгоритму FAST

Висновки до розділу

В даному розділі розглянуті алгоритми та методи порівняння зображень. В загалом, методи порівняння зображень можна поділити на методи, що генерують хеш-код та виконують порівняння його значення з іншим, та методи, що базуються на визначенні ключових точок (ознак) зображення.

Для генерування хеш-коду зображень необхідно зменшити вхідне зображення до певних розмірів. Оптимальний розмір зображення вибирається експериментально та залежить від конкретної мети та необхідної точності порівняння. Зменшення розміру дозволяє позбавитися від високих частот, що визначають деталізацію зображення. Результуюче зображення переводиться у відтінки сірого та є аргументом хеш-функції, результатом якої є хеш-код. Хеш-коди можуть бути порівняні між собою за допомогою відстані Геммінга.

Методи визначення ознак зображення замінюють зображення деякою моделлю – набором ключових точок цього зображення. Ключова точка - це точка зображеного об'єкта, яка з великою часткою ймовірності буде знайдена на іншому зображенні цього ж об'єкта. Детектором є алгоритм виявлення точок інтересу на зображенні, а дескриптор – її ідентифікатор, що виділяє цю точку серед багатьох інших точок. Дескриптори індексують, зберігаючи у спеціальні структури даних, що виконують ефективний пошук.

Методи генерування хеш-коду є простими у реалізації та ефективними у плані швидкодії. Перевагою таких методів є те, що вони можуть порівнювати зображення, навіть якщо одному з них змінили розмір або співвідношення сторін. Алгоритм pHash має гарні дискримінаційні характеристики за рахунок використання ДКП. Однак, один із головних недоліків алгоритмів обчислення хеш-кодів є те, що у разі повороту зображення такі алгортми не можуть

зіставити повернуте зображення з оригінальним. Найкращі показники при повороті зображення у алгоритма pHash, який частково інваріантний до повороту зображення до 3 градусів. Також такі алгоритми є чутливими до обертання, обрізання зображень.

Методи виявлення ознак, своєю чергою, дозволяють отримати набір ключових точок, за якими відбувається порівняння. Це дозволяє розглядувати зображення не як одне ціле, а як набір окремих ознак. За цими ознаками можливе порівняння зображень та класифікація об’єктів на зображеннях.

Використання конкретного алгоритму залежить від ситуації та задач, що перед ним поставлені. Порівнюючи однотипні несильно модифіковані зображення доцільно використовувати алгоритми порівняння через хеш-код, адже вони прості в реалізації, інтеграції та впровадженні в систему. Чим простіше рішення – тим менша ймовірність його помилок, збоїв та некоректної роботи, а також їх відстеження та виправлення. При відомій структурі зображень, можна використовувати такі прийоми, як обрізання зображень або його розбивання на декілька частин, що дозволить визначити найбільш інформативно значущі частини цих зображень.

2. ПЕРЦЕПТИВНІ ХЕШ-ФУНКЦІЇ ТА ЇХ ВИКОРИСТАННЯ

2.1 Поняття медіа-об’єкту та операції над медіа-об’єктами

Через постійно зростаючу цифровізацію аутентифікація мультимедійного вмісту стає все більш важливою. Аутентифікація в цілому означає відповідь на питання про автентичність об’єкта чи ні. Тобто, якщо деякий об’єкт збігається з заданим оригінальним об'єктом.

Аутентифікація сильно залежить від типу об’єкта. Під час автентифікації виконуваного файлу важливо, щоб кожен окремий біт точно відповідав вихідному виконуваному файлу. Для ров’язання таких задач використовують функції криптографічного хешу. Для перевірки автентичності мультимедійного вмісту краще підходять інші методи. Мультимедійний об’єкт, наприклад зображення, може мати різні цифрові аналогічні або схожі зображення, які всі виглядають однаково зі сторони сприйняття людиною. Різні цифрові подання можуть бути результатом таких етапів обробки зображень, як обрізання, стиснення або вирівнювання гістограми. Кожен із цих кроків обробки зображень змінює двійкове представлення зображення. Тому використання криптографічної хеш-функції для аутентифікації модифікованих зображень не працює для мультимедійного контенту. Для визначення "рівності" мультимедійного вмісту пропонуються так звані перцептивні хеш-функції. В останні роки спостерігається зростання наукового та промислового інтересу до технологій перцептивного хешування. Такі функції були розроблені для різних типів цифрових носіїв (наприклад, аудіо, зображення або відео). Перцептивні хеш-функції визначають певні особливості з мультимедійного вмісту та обчислюють хеш-значення на основі цих особливостей. Під час автентифікації мультимедійного об’єкта

порівнюються хеш-значення початкового об’єкта та об’єкта, що підлягає автентифікації за допомогою специфічних функцій. Такі функції обчислюють оцінку відстані або подібності між двома перцептивними хеш-значеннями. Остаточний вердикт ґрунтується на обраному пороговому значенні.

У даній роботі загальний термін "медіа-об'єкт" використовується для мультимедійного вмісту, такого як аудіо, зображення або відеофайлу. Медіа-об’єкт можна змінити, використовуючи різні операції. Прикладом операції із зображенням є обрізання зображення на 10%. Операція - загальний термін для модифікації або маніпуляції.

Модифікація – операція, яка не змінює суттєвого змісту медіа-об'єкта. Після модифікації медіа-об'єкт все ще очікується виявити автентичним за допомогою функції перцептивного хешу.

Маніпуляція - операція, яка суттєво змінює вміст медіа-об'єкта.

Після маніпуляції медіа-об'єкт, як очікується, буде виявлений як неавтентичний за допомогою перцептивної хеш-функції.

2.2 Визначення хеш-функції та її властивості

Для розуміння хеш-функцій та взаємозв'язку перцептивних хеш-функцій, наприклад криптографічних хеш-функцій, надамо загальне визначення хеш-функції. На найвищому рівні хеш-функції можна класифікувати на два типи: ключові хеш-функції та неключові хеш-функції [15, с. 322]. Неключова хеш-функція генерує значення хеш-коду з довільного вводу. Ключова хеш-функція генерує хеш-код з довільного вводу та секретного ключа деякого секретного ключа. Ключові хеш-функції також

називаються кодами автентифікації повідомлень (MAC). Ми обмежуємо свою увагу неключовими хеш-функціями:

Хеш-функцією є функція H, яка має, як мінімум, дві наступні властивості[15, с. 322]:

- стиснення – хеш-функція H перетворює вхідний об’єкт x довільної кінцевої бітової довжини на результат H(x) визначеної бітової довжини n;
- простота обчислення – дана хеш-функція H та вхідний об’єкт x, H(x) – функція, проста в обчисленні комп’ютером.

Нехай P позначає ймовірність. Нехай H позначає хеш-функцію, яка приймає в якості вхідного сигналу один медіа-об'єкт (наприклад, зображення) і створює двійковий рядок довжиною *l*. Нехай *x* позначає конкретний медіа-об'єкт, а x_{mod} означає модифіковану версію цього медіа-об'єкта, яка "сприймається схожою" на *x*. Нехай *y* позначає медіа-об'єкт, який "перцептивно відрізняється" від *x*. Нехай x' та y' позначають хеш-значення. $\{0/1\}^{l}$ представляє двійкові рядки довжиною *l*. Тоді дамо математичне визначення наступним поняттям (властивостям) [16, розділ 1][17, розділ 2][18, розділ 2.2]:

- рівний розподіл хеш-значень (2.1):

$$\mathrm{P}(H(x) = x`) \approx \frac{1}{2^{l}}, \forall x` \in .\{0/1\}^{l} \qquad (2.1)$$

- парна незалежність між персептивно різними медіа-об’єктами *x* та *y* (2.2):

$$\mathrm{P}(H(x) = x`|H(y) = y`) \approx \mathrm{P}(H(x) = x`), \forall x`, y` \in \{0/1\}^{l} \qquad (2.2)$$

− інваріантність перцептивно подібних медіа-об'єктів x і x_{mod} (2.3):

$$\mathrm{P}(H(x) = H(x_{mod})) \approx 1 \qquad (2.3)$$

− різниця перцептивно різних об’єктів x та y (2.4):

$$\mathrm{P}(H(x) = H(y)) \approx 0 \qquad (2.4)$$

Щоб задовольнити властивість (2.3), алгоритми розроблюються з метою визначення особливостей медіа-об’єктів, які інваріантні при незначних глобальних модифікаціях. Для зображень такими глобальними модифікаціями є, наприклад, стиснення чи обрізання. Властивість 2.3 також означає, що для даного медіа-об'єкта x майже неможливо побудувати різний у сприйнятті медіа-об'єкт у такий, що H(x) = H(y) [17, розділ 1].

Проблема при розробці перцептивних хеш-функцій полягає в тому, що автентичні медіа-об'єкти не можна точно відокремити від неавтентичних. Щоб краще зрозуміти цю проблему, наведено наступний приклад. The Joint Photographic Experts Group (JPEG) - це операція із зображенням, яка зазвичай не змінює зображення у сприйнятливому значенні. Тобто застосування стиснення JPEG до зображення не повинно робити його неавтентичним. Проте стиснення JPEG, особливо при застосуванні з використанням низькоякісних параметрів, може значно розмити зображення. Тому особливо зображення, що містять дрібні деталі, важливі для їх семантичного значення (наприклад, зображення, що містить дорожні знаки та номерні знаки автомобілів), можуть сильно постраждати при застосованні стиснення JPEG і повинно бути визначеним неавтентичним. Автори «First summary report on authentication»

[18] узагальнюють це наступним чином: «Для деяких операцій з обробки важко визначити, чи результат модифікацій є автентичним. На кінцевий вердикт також впливають сценарії застосування».

У «Proceedings of the International Conference on Image Processing» [19] пропонується визначене тлумачення автентичності: «Зображення, яке по бітах ідентичне оригіналу, вважається повністю автентичним (міра автентичності 1.0). Зображення, яке не має нічого спільного з оригінальним зображенням, вважатиметься не автентичним (міра автентичності 0.0). Усі інші зображення є частково автентичними. Частково автентичні зображення - це нечітко визначена концепція, і вимірювання автентичності є суб'єктивним і змінюється в залежності від домену програми».

Даний показник автентичності можна проілюструвати як криву автентичності проти модифікації. Для кожного різного типу модифікації визначається відповідна крива. Рисунок 2.1 ілюструє приклад такої кривої достовірності та модифікації. Крива "JPEG Compression" співвідносить автентичність зображення з коефіцієнтом якості використовуваного стиснення JPEG. Крива "Обрізання" співвідносить автентичність із відсотком залишкових пікселів після обрізання зображення. Використовуючи порогове значення, автентичність можна виміряти у двійковій величині.

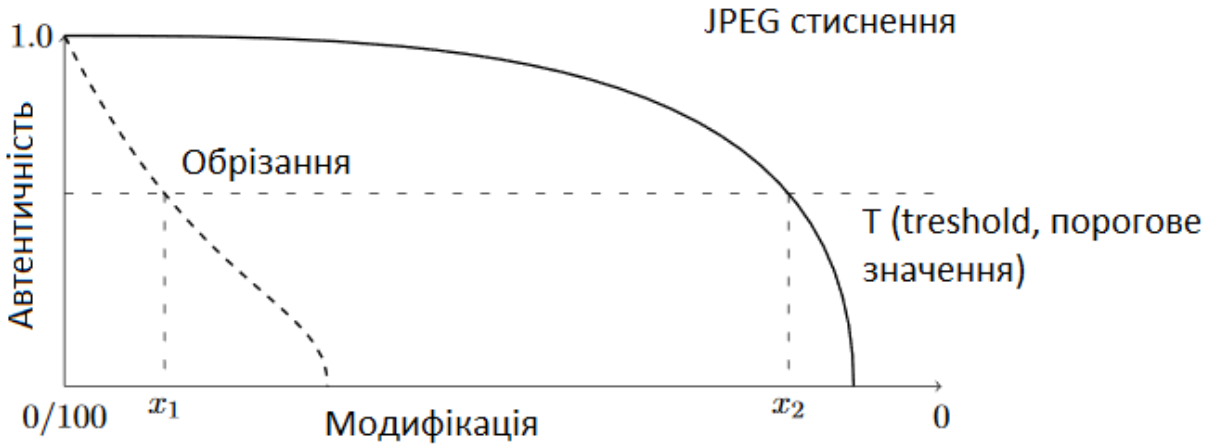

Рисунок 2.1 – Крива автентичності до модифікації.

Відповідно до рисунка 2.1, обрізання зображення на 100 відсотків призведе до того, що зображення буде неавтентичним. Те саме стосується випадку, коли зображення стискається за допомогою параметра якості JPEG між x2 та 0. Таким чином, крива достовірності та модифікації повинна мати похилий нахил для модифікацій та крутий для маніпуляцій.

Фактичними застосуваннями перцептивних хеш-функцій є виявлення спаму зображення, пошук в Інтернеті порушень авторських прав або ведення баз даних незаконного вмісту. Криміналістичні програми, такі як EnCase або Forensic Toolkit, використовують лише криптографічні хеш-функції для індексації та пошуку файлів. Перцептивні хеш-функції були б доцільним доповненням до цих програм.

2.3 Використання перцептивних хеш-функцій

Незважаючи на різні сценарії додатків, які використовують перцептивні хеш-функції, можна вивести різні загальні «моделі використання» [20, розділ 2.1.2]:

- ідентифікація контенту;
- перевірка цілісності;
- підтримка водяних знаків;
- медіа парсинг на основі контенту та процесінг;

Перцептивні хеш-функції є ефективними для пошуку у великих базах даних необхідного мультимедійного вмісту. Наприклад, у роботі «An efficient database search strategy for audio fingerprinting» [21] пропонується функція хешування перцептивного аудіо та дуже ефективна стратегія пошуку, яка дозволяє знаходити перцептивний хеш-код аудіо файлів при роботі з великою базою даних. Використання перцептивних хеш-функцій для таких додатків також означає, що у базі даних потрібно зберігати лише значення хешу та відповідні метадані (наприклад, ім'я файлу). Нема потреби зберігати самі мультимедійні об'єкти в базі даних. Це значно зменшує розмір бази даних. І звичайно, ще одна перевага полягає у тому, що при модифікації медіа-об’єкта незначним чином, він все ще може бути знайдений у базі даних. Як обговорювалося раніше, при розробці перцептивної хеш-функції, оптимізована для цього домену використання, необхідно знехтувати властивістю (2.3) на користь (2.2).

Розглянемо приклад такого використання хеш-функції. Він поділяється на дві фази: “створення бази даних” та “ідентифікація вмісту”. На етапі створення бази даних вона заповнюється перцептивними хеш-значеннями

медіа-об'єктів, які слід розпізнавати пізніше. Зазвичай додаткові метадані кожного медіа-об'єкта зберігаються з його хеш-значенням. Це може бути ім'я файлу мультимедійного об'єкта, його тег ID3, якщо це звуковий файл, або теги формату файлу Exif, якщо це файл зображення. На етапі ідентифікації вмісту системі представляється неідентифікований медіа-об'єкт. Медіа-об'єкт обробляється для отримання перцептивного хеш-коду. Потім перцептивний хеш-код порівнюється зі значеннями хеш-кодів, що зберігаються в базі даних. Якщо є збіг, система надасть додаткову інформацію про заздалегідь невпізнаний медіа-об'єкт (доступні метадані, показник надійності збігу і т.д).

База даних вже заповнена під час фази “популяції баз даних”. Тепер він включає перцептивні хеші та відповідні метадані медіа-об'єктів. Для обчислення перцептивного хешу та визначення „збігу” використовується наступний алгоритм (рисунок 2.2):

1) Визначення ознак та їх обробка: Зазвичай мультимедійний вміст повинен бути попередньо оброблений для використання перцептивної хеш-функції. У випадку з зображенням такими необхідними кроками попередньої обробки може бути зменшення розміру зображення до заданої роздільної здатності та/або зведення зображення у відтінки сірого. Далі з медіа-вмісту витягуються ознаки, необхідні для моделювання перцептивного хешу;
2) Моделювання перцептивного хешу: Перцептивний хеш обчислюється за допомогою ознак, визначених на попередньому кроці;
3) Пошук в базі даних: Для порівняння двох перцептивних хеш-кодів слід використовувати спеціальні алгоритми пошуку та функції відстані / подібності відповідно до використовуваної перцептивної хеш-функці;

4) Перевірка гіпотез: На основі заздалегідь визначеного порогу визначається збіг. Тому визначення порогового значення є важливим та залежить від конкретної мети.

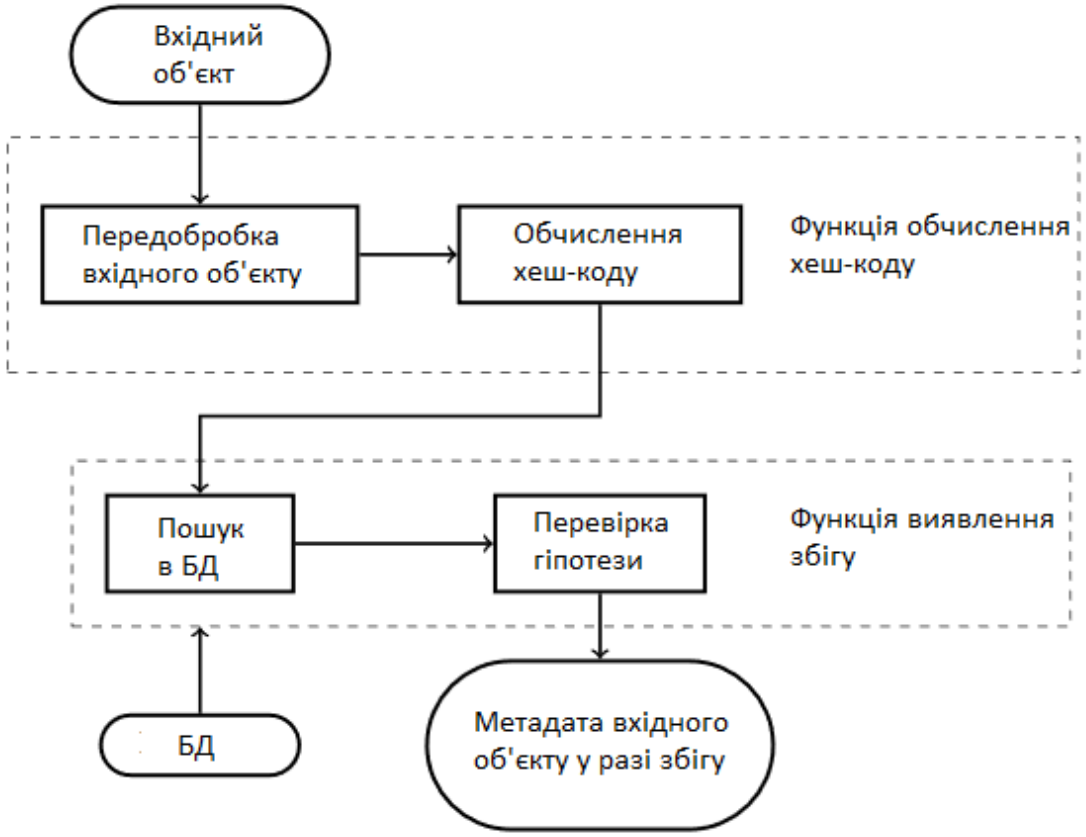

Рисунок 2.2– Загальна схема алгоритму ідентифікації контенту за допомогою хеш-коду

Такий алгоритм може використовуватися також і для перевірки цілісності контенту [19]. Також, хеш-код може бути інтегрований в мультимедійний контент використовуючи цифрові водяні знаки (watermarks). Цифровий підпис (digital signature) може бути використаний для підпису хеш-коду. Цифровий підпис корисний не тільки для автентифікації контенту. Разом з зашифрованою відміткою часу цифровий підпис використовують для верифікації авторських прав. Водяний знак корисний для верифікації

оригінальності медіа-об’єкту. В той же час, цифровий водяний знак не є самодостатнім для верифікації авторського права, тому що медіа-об’єкт може бути позначений декількома водяними знаками. В загалому, цифровий підпис використовується для забезпечення безпеки отримувача (користувача) медіа-об’єкту, а цифрові водяні знаки – для забезпечення авторського права. На рис. 2.3 зображена абстрактна архітектура програмного забезпечення для створення цифрового підпису, а на рисунку 2.4 – його верифікації.

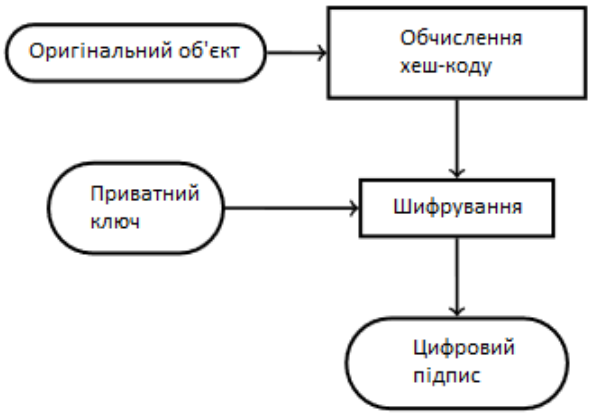

Рисунок 2.3 – Створення цифрового підпису

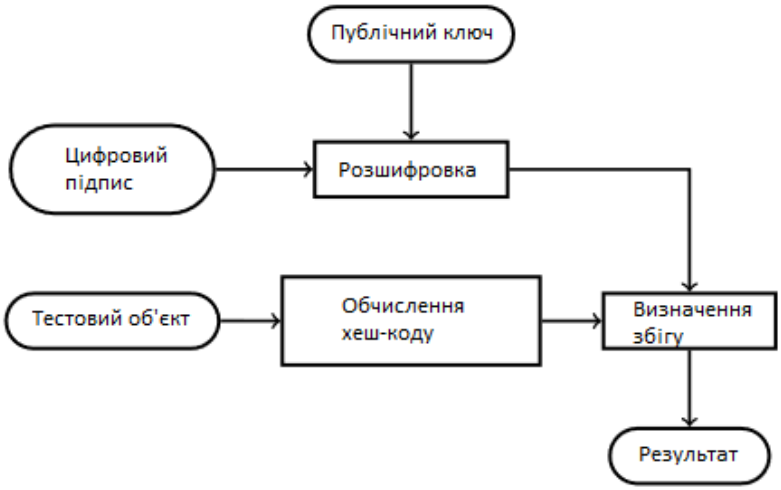

Рисунок 2.4 – Верифікація цифрового підпису

2.4 Функції порівняння хеш-кодів

Перцептивна хеш-функція обчислює подібні перцептивні хеш-значення для подібних медіа-об'єктів. Для порівняння двох перцептивних хеш-кодів слід використовувати відповідні функції визначення міри їх схожості. Найчастіше використовуються Bit Error Rate (BER), Hamming Distance та Peak of Cross Correlation (PCC). Перші два методи вимірюють відстань між двома хеш-значеннями, тоді як PCC вимірює подібність двох хеш-значень.

Bit Error Rate (BER) визначає BER як значення i бітових помилок хеш-коду, нормалізованого за довжиною цього хеш-коду (2.5) [1]:

$$p \coloneqq \frac{i}{k} \tag{2.5}$$

де $i \in \{0, 1, \dots, k\}$ та $0 \leq p \leq 1$.

Кількість бітових помилок I дорівнює відстані Геммінга перцептивних хеш-кодів. При порівнянні перцептивно різних зображень BER повинен становити приблизно 0.5.. Перцептивно рівні зображення повинні давати BER, близький до 0.

Відстань Гемінга є мірою різниці двох рядків [4, с. 154]. Такими рядками можуть бути двійкові закодовані числа, але вони також можуть складатися з елементів інших систем числення. Приклад наведено у таблиці 3.

Таблиця 3 – Приклад обчислення Відстані Хемінга для значень, записаних в двійковій та десятковій системах

Рядок 1	Рядок 2	Відстань Хемінга
00101	10101	1
12345	13344	2

Нехай A позначає алфавіт кінечної довжини. X = (x_1, ...,x_n) позначає рядок парної довжини, тоді як x ∈ A. Те саме справедливо і для y = (y_1, ..., y_n). Тоді відстань Геммінга Δ між x та y визначається як (2.6):

$$\Delta(x, y) \coloneqq \sum_{x_i \neq y_i} 1, i = 1, \dots, n \tag{2.6}$$

Для більш ефективного порівняння, відстань Гемінга може бути нормалізована щодо довжини рядків (2.7) [5]:

$$\Delta_n(x, y) \coloneqq \frac{1}{n} \sum_{x_i \neq y_i} 1, i = 1, \dots, n \tag{2.7}$$

Для обчислення відстані Гемінга двійкових закодованих чисел може використовуватися операція XOR. Нехай a і b позначають два двійкові закодовані числа однакової довжини. Тоді відстань Хемінга дорівнює кількості одиниць у результаті обчислення a⊕b.

Іншою метрикою, яку можна використовувати, є Equality Percentage (EP) (2.8), що запропонована у роботі [18]:

$$EP \coloneqq 100 * \Delta_n \tag{2.8}$$

Для перцептивно подібних зображень EP має приймати великі значення (≈100%). І навпаки, для перцептивно різних зображень EP має приймати малі значення (≈0%). Знову ж таки, очікуване значення EP для двох перцептивних хеш-значень, отриманих з рівномірного випадкового розподілу $\{0,1\}^n$, становить приблизно 50%.

Peak Of Cross Correlation (PCC) між двома сигналами визначається як (2.9) [18]:

$$r_{xy}(T) = \int_{-\infty}^{\infty} x(t)\, y(t+T)dt \tag{2.9}$$

де x (t) та y (t) - дві детерміновані, реальні функції. Функція кореляції $r_{xy}(T)$ описує збіг цих двох сигналів щодо часу зсуву T. Значення T визначає, наскільки другий сигнал зміщений вліво.

Маємо два ряди x_i та y_i, де i = 0, 1, 2, …, N – 1 та N позначає довжину двох рядів, тоді нормалізоване значення PCC r та затримка d визначаються як (2.10):

$$r_d = \frac{\sum_i [(x_i - mx) * (y_{i-d} - my)]}{\sum_i \left[(x_i - mx)^2 * \sqrt{(y_{i-d} - my)^2}\right]} \tag{2.10}$$

Висновки до розділу

У даному розділі розглянуті визначення медіа-об’єкту, операцій над ним, визначення та властивості хеш-функцій, а також функції порівняння хеш-кодів.Визначено поняття перцептивної хеш-функції, яка дозволяє математично ідентифікувати медіа-об’єкт. Для розв’язання певних задач, хеш-функція повинна забезпечувати хеш-код, значення якого інваріантне до різного роду модифікацій.

Розглянуто моделі використання перцептивних хеш-функцій та хеш-кодів. Перцептивні хеш-функції є ефективними для пошуку у великих базах даних необхідного мультимедійного вмісту. Нема потреби зберігати самі мультимедійні об'єкти в базі даних. Достатньо зберігати обчислені хеш-значення та, за потреби, метадані про медіа-об’єкт. Це значно зменшує розмір бази даних. І звичайно, ще одна перевага полягає у тому, що при модифікації медіа-об’єкта незначним чином, він все ще може бути знайдений у базі даних.

Для порівняння хеш-кодів можуть використовуватися функції Bit Error Rate (BER), Hamming Distance та Peak of Cross Correlation (PCC). Відстань Гемінга має широке використання та є мірою різниці двох рядків, що порівнюються посимвольно (або побітово). Bit Error Rate дозволяє представити значення відстані Гемінга у межах від 0 до 1.

Слід зазначити, що розглянуті методи можна використовувати також для порівняння томографічних знимків

3. РОЗРОБКА АЛГОРИТМУ ПОРІВНЯННЯ ТОМОГРАФІЧНИХ ЗНІМКІВ

3.1 Підстава для розробки

Важливим є порівняння томографічних знімків. Такі знімки мають визначену структуру та подаються в одному форматі. Порівняння томографічних знімків різних органів (наприклад, мозку та легенів) не є доцільним та не несе практичної цінності. Різні типи томографічних знімків завідомо відрізняються, тому можемо стверджувати про відсутність критеріїв оцінки схожості таких знімків.

Аналізуючи набір томографічних знімків, доцільно визначити особливості, якими вони можуть відрізнятися, а також перелічити типи модифікацій, до яких необхідно забезпечити інваріантність [22]. Для томографічних знімків необхідно забезпечити інваріантність до:

1) Зміни вмісту томографічних знімків. Томографічні знімки відрізняються між собою в залежності від пацієнта, стану його здоров’я та прогресу лікування;
2) Обертання зображення на невеликий кут повороту (приблизно до 5 градусів є достатнім для вирішення практичних задач). Зображення не завжди ідеально рівні відносно горизонтальної осі. Приклад таких зображень наведено на рис. 3.1. Таке обертання знімку може бути, наприклад, результатом різного положення тіла пацієнтів під час виконання комп’ютерної томографії. Знімки не можуть бути зроблені ідеально. Оскільки обертання зображення не є об’єктом аналізу, доцільно, щоб обертання зображень критично не впливало на результат порівняння.

3) Стиснення зображень. Інваріантість до стиснення може бути корисною, якщо алгоритм використовується в рамках більшого програмного комплексу або відповідно до певних потреб;
4) Зміни розміру зображень.
5) Модифікації зображення шляхом коригування його різкості. Коригування різкості використовується для медичних зображень з метою покращення їх візуальної інформативності, що важливо при аналізі знімків лікарями;
6) Зміни яскравості зображення для кращої візуалізації медичних знімків.

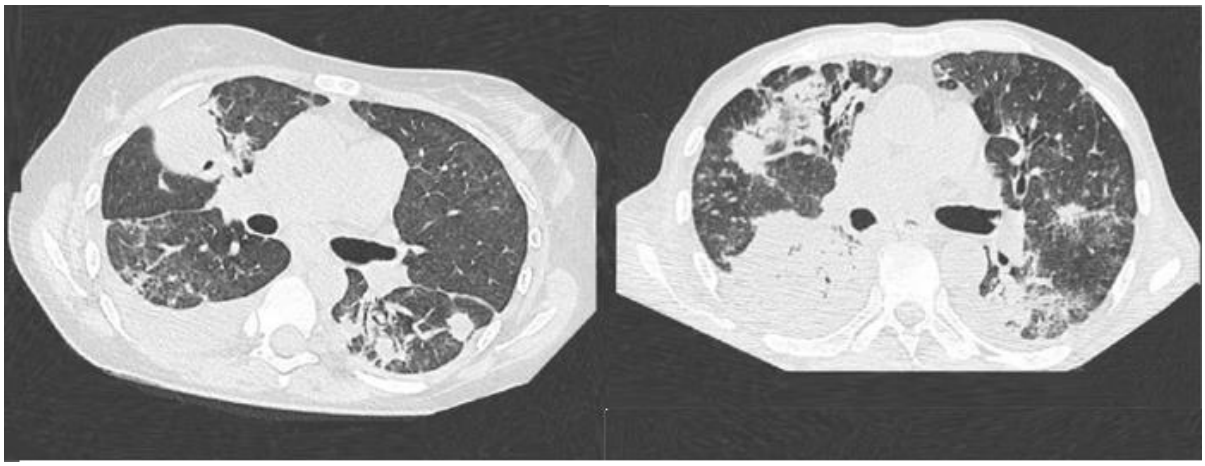

Рисунок 3.1 – Приклад томографічних знімків. Зображення зліва є оберненим на невеликий кут відносно зображення зправа

Відповідно до першого розділу, алгоритми порівняння зображень умовно діляться на два типи:

1) Алгоритми порівняння за допомогою хеш-коду;
2) Алгоритми порівняння за ознаками та визначенням ключових точок.

Другий тип алгоритмів не є необхідним для порівняння томографічних зображень. Такі алгоритми використовуються для визначення конкретних ознак зображень, що їх ідентифікують. Тобто, такі алгоритми використовують для зіставлення зображень з різним змістом, а також для виявлення конкретних об’єктів на зображенні. Томографічні знімки є визначеними за типом. Тобто, метою та практичною цінністю цієї роботи є зіставлення двох однотипних томографічних зображень та визначення міри їх схожості.

Розглянуті алгоритми першого типу є доцільними для зіставлення томографічних знімків. Такі алгоритми є простими та забезпечують інваріантність до модифікацій, наведених вище, без додаткових затрат та модифікацій. Але такі алгоритми не є інваріантними до обертання зображень. Так, один із найбільш поширених перцептивних алгоритмів pHash є інваріантним до обертання зображення лише на невеликий кут повороту (до 2-3°). Інші алгоритми не показують кращі результати. Тому, для запезпечення інваріантності (часткової інваріантності) до обертання зображень необхідно використовувати інший алгоритм, що може базуватися на перцептивних алгоритмах, використовуючи всі їх переваги.

3.2 Алгоритм Block Truncation Coding та його особливості

Основоположним джерелом ідеї реалізації нового алгоритму слугує робота дослідника Edward J. Delp’a з університету Пердью. У 1979 році побачила світ публікація автора під назвою «Image Compression Using Block Truncation Coding». В даній публікації запропонований метод стиснення

зображень шляхом його розбиття на N блоків та обробки пікселів кожного блоку окремо. Алгоритм отримав назву «Block Truncation Coding» [23].

Block Truncation Coding (BTC) - це метод стиснення зображень з втратами для зображень у градаціях сірого. Алгоритм ділить вхідне зображення на блоки, а потім використовує квантор, щоб зменшити кількість рівнів сірого в кожному блоці, зберігаючи однакове середнє та середньоквадратичне відхилення. Даний алгоритм - ранній попередник популярної апаратної техніки DXTC стиснення текстур, хоча метод стиснення BTC вперше був адаптований до кольору задовго до DXTC, використовуючи дуже подібний підхід, який називається Color Cell Compression [24]. BTC також адаптований до стиснення відео. Скорочено, алгоритм складається з наступних кроків:

1) Звести зображення до градацій сірого;

2) Розбити вхідне зображення на N блоків 4x4;

3) Обчислити середнє значення пікселів кожного блоку (3.1):

$$I` = \frac{1}{16}\sum_{i=0}^{15} I(i) \tag{3.1}$$

4) Обчислити середнє квадратичне відхилення кожного блоку (3.2):

$$\sigma = \sqrt{\frac{1}{16}\sum_{i=0}^{15}(I(i) - I`)^2} \tag{3.2}$$

5) Значення кожного пікселю блоку заміняється на значення 0 або 1, в залежності від того, чи його значення більше чи менше за середнє (3.3):

$$b(i) = \begin{cases} 1; I(i) \geq I` \\ 0; I(i) < I` \end{cases} \quad (3.3)$$

Тобто, необхідно лише 16 біт, щоб закодувати блок розміром 4x4, що в свою чергу складається з 16 значень пікселів відтінків сірого.

Наприклад, для того, щоб закодувати блок, вказаний на рис. 3.1, нам необхідно обчислити середнє значення (в даному випадку седенє значення становить 150), середнє квадратичне відхилення (74) та бінарну матрицю, що вказує, чи кожен піксель має значення більше чи менше за середнє.

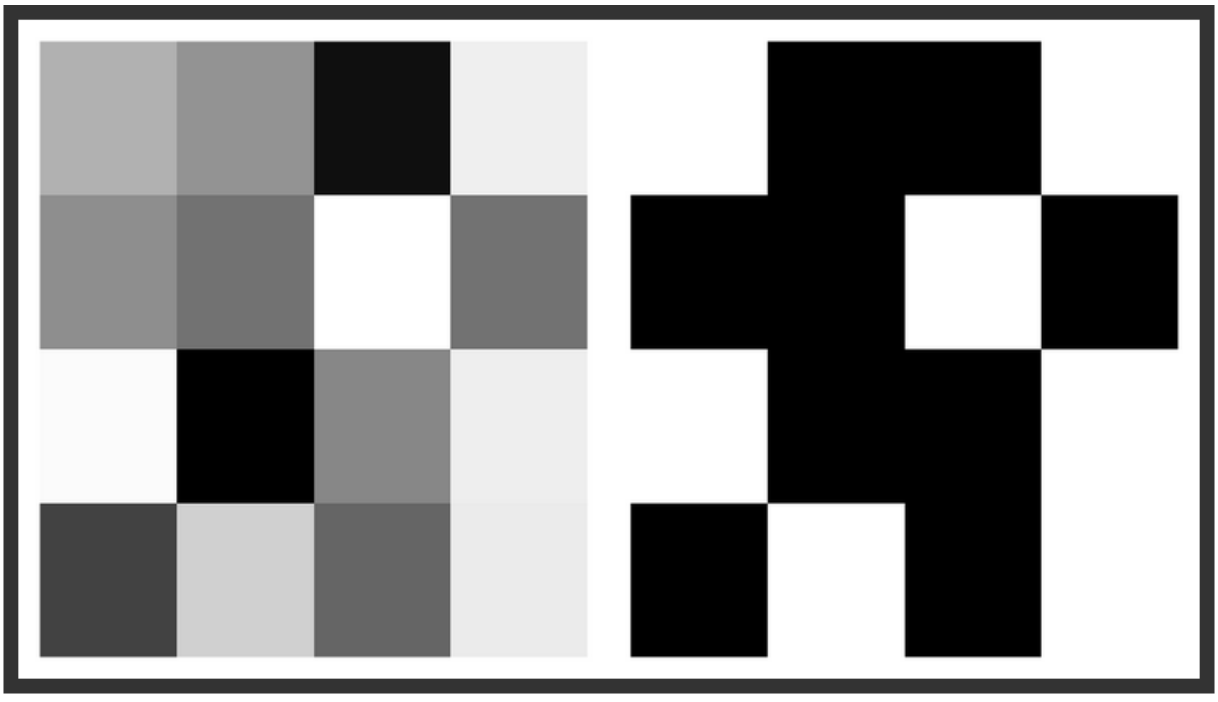

Рисунок 3.1 – Приклад блоку 4x4, що є результатом розбиття вхідного зображення на блоки цього розміру (зліва) та закодованого блоку, що складається з бінарних значень (зправа)

Закодований блок можливо декодувати, обчисливши кількість нулів та одиниць та маючи значення середньоквадратичного відхилення. Дана інформація допоможе побудувати гістограму розподілу оригінальних значень пікселів.

Приклад стисненого зображення наведено на рис. 3.2. Тестові зображення взяті з відкритого ресурсу visualhunt.com [25].

Розбиття зображення на блоки може використовуватися для вирішення різних типів задач. Так, операції розбиття зображення на блоки та обчислення їх середнього значення входять до стандартного пакету MatLab. Прикладом такої функції є blkproc(img, [M N], 'mean2').

Рисунок 3.2 – Приклад стисненого зображення алгоритмом BTC. Оригінальне зображення зліва, стиснене – справа.

Обчислення середнього значення блоку – процес обчислення середнього значення значень всіх пікселів блоку. Блоки не перетинаються. Результатом такого обчислення є єдине середнє значення, що заміняє всі значення пікселів блоку.

Алгоритм є простим у реалізації та ефективним по швидкодії. Стиснене зображення втрачає якість, що легко помітити візуально, порівнюючи оригінальне та стиснене зображення. Для свого часу даний алгоритм був ефективним та доцільним рішенням. Використання його при теперішніх конфігураціях ЕОМ не є доцільним. Однак, з даного методу є доцільним використання його переваг. Для даної роботи корисними можуть бути дві властивості:

1) Простота обчислень та швидкодія. Розбиття зображення на блоки та обчислення середніх значень не є ресурсозатратним;
2) Розбиття зображення на блоки. Це дозволить оцінювати зображення частинами.

Алгоритм Average Hash використовує аналогічні прості обчислення середнього значення. Але, даний алгоритм розцінює зображення як одне ціле.

Тобто, блокування зображення може бути корисним для модифікації алгоритму Average Hash, що дозволить зберегти швидкодію алгоритму. Гіпотетично, блокування зображення дозволить отримати кращі дискримінаційні характеристики алгоритму. Також, даний метод забезпечить деяку інваріантність до обертання зображення. Це є результатом того, що у результаті розбиття блоки можуть містити однотонні області з мінімальними різницями яскравості. Тобто, обертання таких областей не матиме великий вплив на результат обчислення хеш-коду зображення. Приклад такої області вказано на рис. 3.3. На рисунку помітно, що область зображення є однотонною

(майже однотонною), та, потенційно, частково інваріантною до обертання. Область на рисунку масштабовано для кращої візуалізації.

Особливо дана властивість може бути корисна для медичних зображень. Наприклад, на рис. 3.4 зображено чотири результати томографічних знімків легенів людини. На даних знімках можливо візуально виділити області, що є однотонними (частково однотонними). Також, такі зображення є чорно-білими за своїм визначенням.

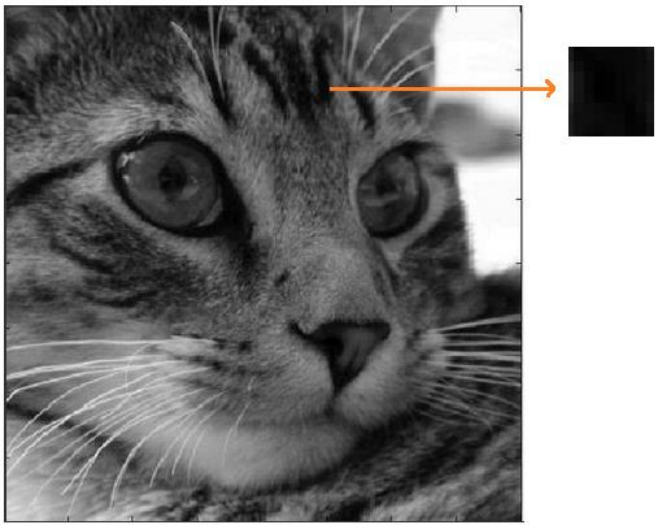

Рисунок 3.3 – Приклад однотонної області зображення

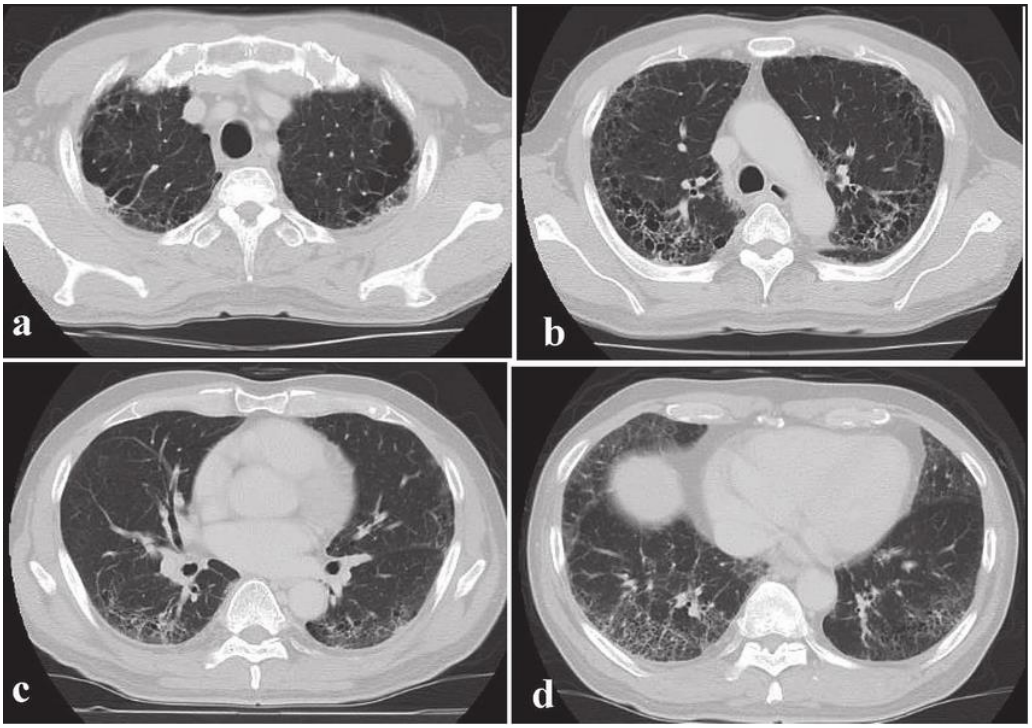

Рисунок 3.4 – Приклади томографічних знімків легенів людини

3.3 Розробка блочного алгоритму обчислення хеш-коду зображення

Виходячи з переваг розбиття зображення на частини, що перечислені у попередньому підрозділі, доцільно розробити алгоритм генерації хеш-коду, використовуючи даний підхід. В цілому, розроблений алгоритм схожий на алгоритм Average Hash, який обчислює середні значення пікселів для генерування хеш-коду. В модифікованому алгоритмі зображення розбивається на N блоків. Далі, для модифікованого алгоритму середні значення рахуються не для всіх значень пікселів (всього зображення), а для пікселів кожного блоку окремо. В результаті маємо N середніх значень для кожного блоку. Для генерування остаточного хеш-коду є можливим

порівняння середніх значень кожного блоку з їх медіанним значенням. Це дозволяє отримати бінарне представлення хеш-коду. Опишемо алгоритм.

Нехай N позначає довжину результативного хеш-коду (в бітах). k – кількість значень пікселів у кожному блоці у результаті розбиття зображення:

1) Зменшити зображення та інтерполювати його. Наприклад, слідуючи BTC алгоритму, можна зменшити розмір до N*4xN*4. Такий розмір зображення та виконання наступних кроків алгоритму визначать N блоків розміром 4x4;
2) Звести зображення у градації сірого. Існують декілька методів для цієї досягнення цієї мети, які перечислені нижче (3.7, 3.8, 3.9). У результаті отримаємо матрицю значень відтінків сірого;
3) Розбити зображення I на N блоків та розгорнути їх у вектори $\{I_1, I_2, \dots, I_N\}$, що не перетинаються. На рис. 3.5 наведено приклад такого розбиття на 16 блоків. На практиці, довжина хеш-коду повинна становити хоча б 64 біти, тому на практиці кількість блоків становить 64 та більше. На 3.6 наведено приклад розгортки блоку 13 у вектор.

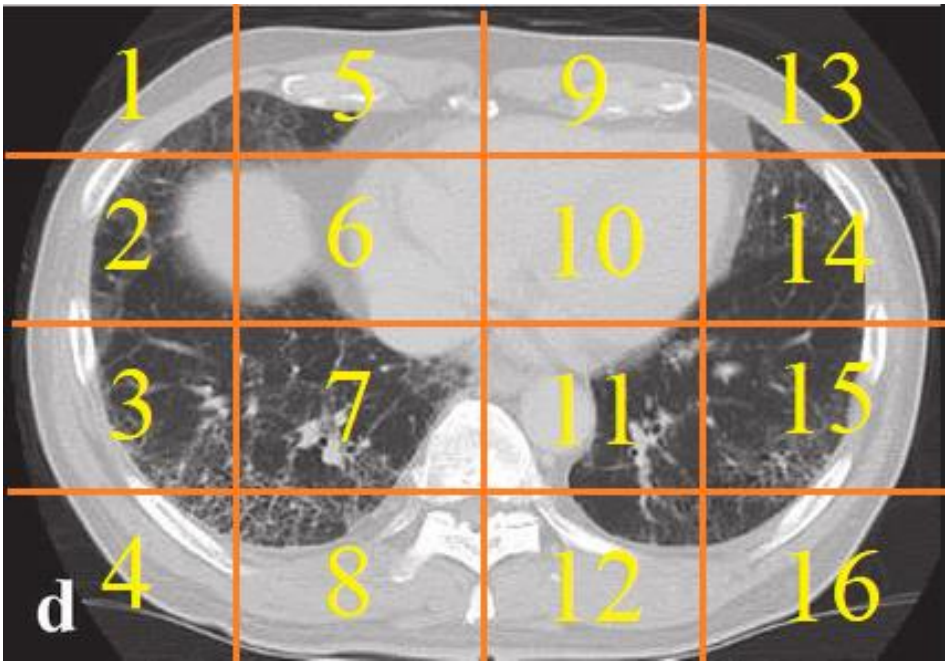

Рисунок 3.5 – Розбиття зображення на 16 блоків. За допомогою такого розбиття можливо згенерувати хеш-код довжиною 16 бітів.

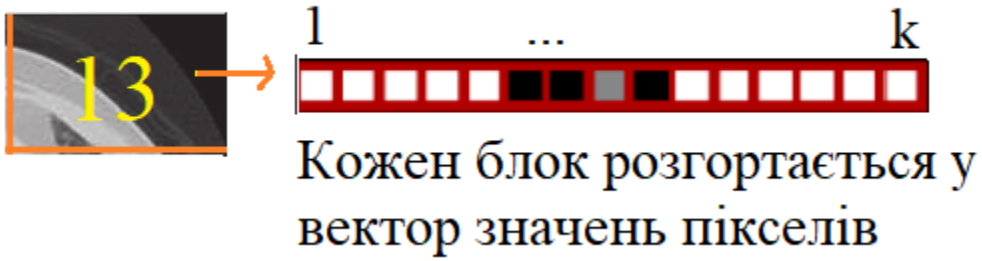

Рисунок 3.6 – Розгортання блоку зображення у вектор. Довжина вектору k залежить від кількості блоків N та розміру зображення

4) Обчислити середнє значення кожного вектора. Тобто, отримаємо послідовність середніх значень $\{M_1, M_2, \dots, M_N\}$ відповідних векторів $\{I_1', I_2', \dots, I_N'\}$ (3.4):

$$M(i) = \frac{1}{k}\sum_{i=0}^{k-1} I'(i) \tag{3.4}$$

5) Обчислити медіанне значення M_d всіх середніх значень $\{M_1, M_2, \dots, M_N\}$. Медіана — це величина ознаки, що розташована посередині ранжованого ряду вибірки, тобто це — величина, що розташована в середині ряду величин у зростаючому або спадному порядку. Медіана ділить ряд значень ознаки на дві рівні частини, по обидві частини від неї розміщується однакова кількість одиниць сукупності. Тобто, вектор значень $\{M_1, M_2, \dots, M_N\}$ сортується за зростанням, а потом знаходиться індекс медіанного значення d за формулою (3.5):

$$d = \frac{N+1}{2} \tag{3.5}$$

6) Звести середні значення $\{M_1, M_2, \dots, M_N\}$ у бінарний вигляд та отримати хеш-код h як (3.6):

$$h(i) = \begin{cases} 0, & M_i < M_d \\ 1, & M_i \geq M_d \end{cases} \tag{3.6}$$

де i = 0, …, N – 1.

У результаті отримаємо бінарний рядок h довжиною N, що є хеш-кодом зображення.

Для забезпечення часткової інваріантності до поворотів зображення пропонується наступний метод:

1) Зменшити зображення до розміру 256x256 та інтерполювати його;

2) Звести зображення у градації сірого;
3) Розділити значення пікселів зображення I на N блоків (векторів) $\{I_1, I_2, \dots, I_N\}$, що не перетинаються (рис. 3.5 та 3.6);
4) Зашифрувати індекси векторів $\{I_1, I_2, \dots, I_N\}$, використовуючи секретний ключ K для того, щоб отримати послідовність векторів з новим порядком сканування $\{I_1', I_2', \dots, I_N'\}$.
5) Обчислити середнє значення значень кожного вектора. Тобто, послідовність середніх значень $\{M_1, M_2, \dots, M_N\}$ відповідних векторів $\{I_1', I_2', \dots, I_N'\}$ (3.4).
6) Обчислити медіанне значення M_d всіх середніх значень $\{M_1, M_2, \dots, M_N\}$ (3.5);
7) Повернути на D градусів матрицю M, отриману зі значень $\{M_1, M_2, \dots, M_N\}$, де D = {0, 15, 30, …, 345}. У результаті отримаємо 24 матриці M_i (i = 1, 2, …, 24).
8) Розбити кону з 24 обернених матриць на N блоків. Визначити середні значення $\{M_{i1}, M_{i2}, \dots, M_{iN}\}$ кожного блоку та медіанне значення M_{di} цієї послідовності. У результаті отримаємо 24 групи послідовностей. Автор не вказує, яку операцію обертання матриці використовувати. Автор цієї пропонує використовувати звичайну операцію обертання зображення, не використовуючи інтерполяцію. Крім того, матриці (зображення) не повинні бути збільшені за допомогою операції обертання, оскільки всі матриці повинні мати однакові розміри.

Застосувати вираз (3.6) для кожної з 24 послідовностей та отримати кінечну матрицю хеш-кодів.

3.4 Зменшення розміру зображення та його інтерполяція

Для зменшення розміру зображення широкого використання набув метод під назвою білінійна інтерполяція [26]. Спочатку розглянемо лінійну інтерполяцію.

Допустимо, задано дві точки на прямій з координатою та b, і вони пов'язані зі значеннями A і B. Також маємо третю точку з координатою x, де a ≤ x ≤ b (рис. 3.7):

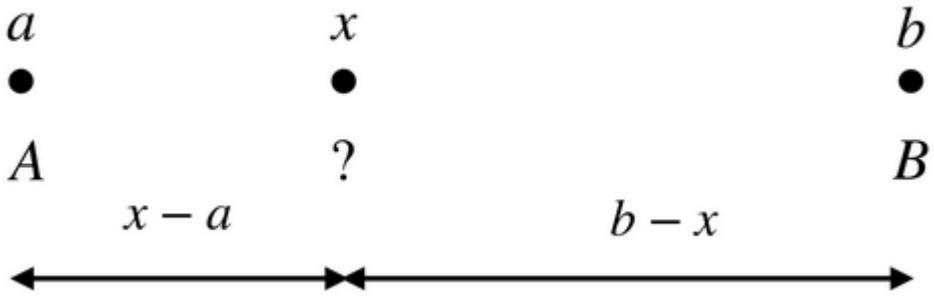

Рисунок 3.7 – Лінійна інтерполяція між двома точками

Лінійна інтерполяція обчислює інтерполяцію як середньозважене значення значень, пов'язаних з двома точками, де ваги пропорційні відстані між x та a, а також x та b (3.7):

$$X = A\frac{b-x}{b-a} + B\frac{x-a}{b-a} \tag{3.7}$$

Вага для коефіцієнта A пропорційна відстані x до b, тоді як вага для B пропорційна його відстані до a. Коли x переходить до a, його значення X стає A; аналогічно, X стає B, коли x переходить до b.

Білінійна інтерполяція може бути розбита на операції лінійної зміни розміру y (висота) та x (ширина). Допустимо, задано чотири точки з координатами (y1, x1), (y1, x2), (y2, x1) та (y2, x2) та пов'язаними з ними

значеннями A, B, C та D. Спочатку обчислюємо інтерпольоване значення X та Y у по ширині (3.8, 3.9):

$$X = A(1 - w_x) + Bw_x \tag{3.8}$$

$$Y = \mathrm{C}(1 - w_x) + Dw_x \tag{3.9}$$

Тоді лінійну інтерполяція між двома інтерпольованими значеннями X та Y у вимірі висоти (3.10):

$$Z = X(1 - w_y) + Yw_y = A(1 - w_x)(1 - w_y) + Bw_x(1 - w_y) + \\ + C(1 - w_x)w_y + Dw_xw_y \tag{3.10}$$

, де $w_x = \frac{x - x_1}{x_2 - x_1}$ та $w_y = \frac{y - y_1}{y_2 - y_1}$

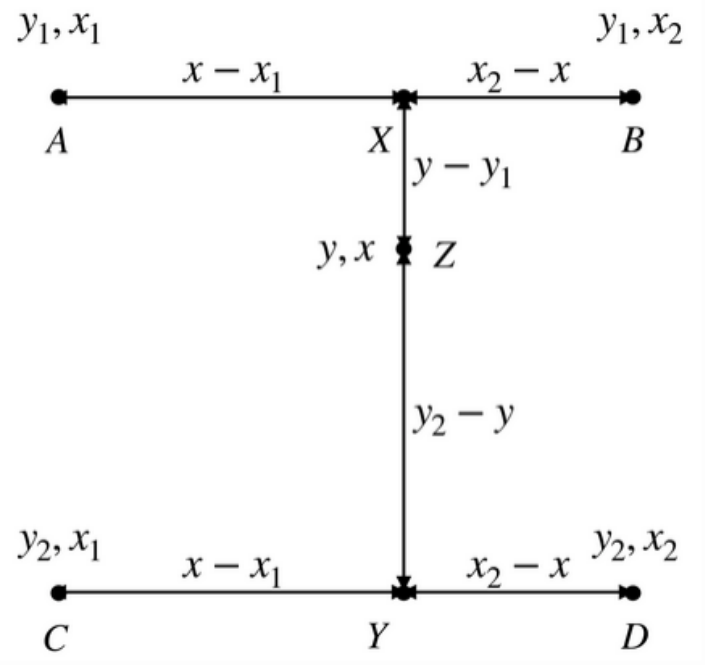

Рисунок 3.8 – Білінійна інтерполяція між чотирма точками

Програмування методу білінійної інтерполяції, описаної вище, дозволить змінити розмір двомірного масиву, яким може бути представлене зображення. Приклад результату виконання такого алгоритму та зміни розміру зображення наведено на рис. 3.9.

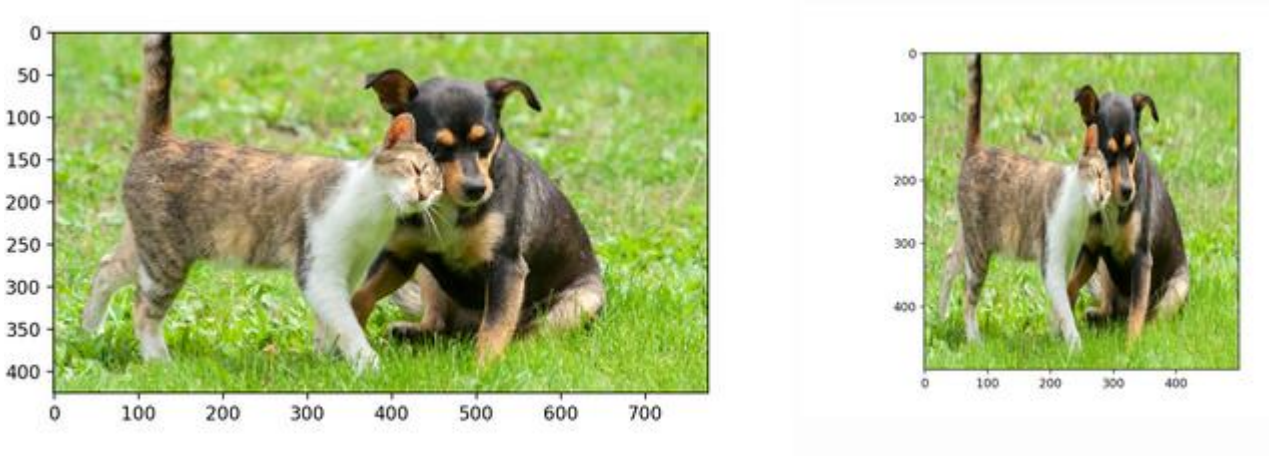

Рисунок 3.9 – Зменшення оригінального зображення розміром 425x755 (зліва) до розміру 500x500 (зправа)

Також широкого використання набув метод бікубічної інтерполяції. Принцип бікубічної інтерполяції схожий з білінійною інтерполяцією, але з парою відмінностей:

1) Білінійна інтерполяція базується на чотирьох найближч их пікселях зображення для визначення результату (тобто, розглядається область 2x2). Бікубічна інтерполяція базується на 16 пікселях (4x4);
2) Розподіл ваги виконується інакше. Ближчі 4 пікселі мають більшу вагу при розрахунках.

Бікубична інтерполяція йде на один крок далі білинійной, розглядаючи масив з 4x4 сусідніх пікселів. Оскільки вони знаходяться на різних відстанях від результативного пікселя, найближчі пікселі отримують більшу вагу при розрахунках. Результатом бікубічної інтерполяції є значно різкіші зображення у порівнянні з білінійною. У більшості випадків бікубічна інтерполяція є оптимальною по співвідношенню часу обробки і якості результату. З цієї причини бікубічна інтерполяція є стандартною для багатьох програм редагування зображень (включаючи Adobe Photoshop), драйверів та камер.

3.5 Методи зведення зображення до градацій сірого

Нехай зображення є кольоровим та подане у RGB. R – значення красного кольору, G – значення зеленого кольору та B - значення чорного кольору. Нехай I – значення відпінку сірого. Для зведення зображення до градацій сірого використовують наступні методи:

a) Метод Lightness [27] усереднює найвизначніші та найменш помітні кольори (3.11):

$$I = \frac{\max(R, G, B) + \min(R, G, B)}{2} \tag{3.11}$$

b) Метод Average [28], що безпосередньо усереднює значення складових R, G та B (3.12):

$$I = \frac{R + G + B}{3} \tag{3.12}$$

c) Метод Luminosity [28], що є модифікацією методу Average. Даний метод також усереднює значення складових R, G та B, враховуючи людське сприйняття. Зір людини більш чутливий до зелених кольорів, тому для зеленого кольору коефіцієнт має найбільше значення (3.13):

$$I = 0.21R + 0.72G + 0.07B \quad (3.13)$$

Результати зображень, перетворених методами нижче, наведені на рис. 3.8, 3.9 та 3.10.

Рисунок 3.8 – Результат застосування «Lightness» методу

Рисунок 3.8 – Результат застосування «Average» методу

Рисунок 3.8 – Результат застосування «Luminosity» методу

Висновки до розділу

У даному розділі розглянуті підстави до створення нового алгоритму для порівняння томографічних знімків. Визначено, що для цієї мети можуть бути використані алгоритми обчислення перцептивних хеш-кодів. Однією з причин до створення нового алгоритму є те, що алгоритми обчислення перцептивних хеш-кодів не є інваріантими до обертання зображень та чутливі до обертання зображень до 2-3°, що сильно впливає на результуючі хеш-коди та результати їх порівняння.

В основі запропонованого алгоритму є алгоритм Average Hash та алгоритм компресії зображень під назвою Block Truncation Coding. Основною ідеєю є розбиття зображень на блоки та обчислення хеш-кодів на основі кожного блоку окремо. Такий підхід є ефективним у плані часу виконання забезпечує часткову інваріантність до обретання зображень.

Також, розглянуті методи інтерполяції зображення та методи зведення зображень у градації сірого. Інтерполяція є важливим кроком при зменшенні розміру зображення, що використовується в алгоритмі. Визначено, що метод бікубічної інтерполяції є оптимальним по співвідношенню часу обробки зображення та якістю результату. Тому використання цього методу інтерполяції є доцільним для роботи алгоритму та модифікації зображень при його тестуванні.

Існує декілька методів зведення зображень до градацій сірого. Оптимальним зі сторони людського сприйняття є метод під назвою «Luminosity». Розроблений алгоритм зводить зображення до градацій сірого для подальших обчислень. Фактор людського сприйняття не є важливим для даного алгоритму. Тому для запропонованого алгоритму доцільно використовувати метод під назвою «Average». Для порівняння томографічних

зображень даний крок не є необхідним, адже такі зображення подаються у чорно-білому вигляді. Але, для подальших досліджень, тестування алгоритму та його аналізу є доцільним використання різних зображень, у тому числі і кольорових.

4. ТЕСТУВАННЯ ТА ДОСЛІДЖЕННЯ ЗАПРОПОНОВАНОГО АЛГОРИТМУ

4.1 Вхідні дані та підходи до тестування алгоритму

Для тестування модифікованого алгоритму використовується набір зображень та розміром 2874 x 2260 [28]. Використано два типи набору зображень:

1) Різнотипні зображення, що мають довільний набір різних зображень;
2) Схожі зображення качок, що плавають у воді.

Кожен набір складається зі ста зображень. Для досліджень визначено набір тестів:

1) Швидкодія;
2) Розподіл значень BER для немодифікованих зображень;
3) Модифікація зображень та їх порівняння з оригінальними;

У тестах використовується розроблений алгоритм та алгоритм pHash, що набув широкого використання та вважається еталонним для вирішення подібного роду задач. Результати запропонованого алгоритму порівнюються з результатами алгоритму pHash та на основі цих порівнянь робляться висновки.

Для перевірки надійності хеш-функцій перцептивного зображення використовуються модифікації зображень. Стиснення JPEG - одна з найбільш поширених операцій з зображеннями. Користувачі виконують стиснення JPEG, щоб, наприклад, зменшити розмір файлу своїх зображень. Операція обертання зображень часто використовується в наукових роботах, щоб продемонструвати стійкість хеш-функцій перцептивного зображення. Також, використовується операція модифікації шляхом повороту зображення на 180°, оскільки дана операція майже не змінює сприйняття людиною зображення та

повинна враховуватися даними алгоритмами. Ще однією типовою модифікацією є зміна розміру зображення.

Над зображеннями проведені наступні модифікації:

1) Зміна розміру зображення;
2) Стиснення JPEG;
3) Обертання зображення на 3° та 5° ;
4) Обертання зображення на 180°.

Якщо метою тесту є вимірювання ефекту операцій модифікації, важливо мати на увазі, що збереження зображення за допомогою певних форматів зображень (наприклад, формат зображення JPEG) є самою модифікацією зображення або маніпуляцією над ним. Зазвичай процес застосування операції до зображення полягає в наступному. Файл зображення зчитується з диска. Потім зображення декодується і зберігається у власному необробленому форматі в системній пам'яті. Згодом застосовується операція (наприклад, поворот зображення). Нарешті, зображення перетворюється із нестандартного необробленого формату в стандартизований формат зображення, а результат записується на жорсткий диск. Для таких досліджень важливо використовувати лише формати зображень, які не використовують стиснення з втратами. Доцільним форматом зображення є, наприклад, PNG. Оскільки формати зображень, що використовують методи стиснення з втратами, не повинні використовуватися для таких тестів, розмір файлів використовуваних наборів зображень збільшується.

4.2 Час виконання

Швидкодія перцептивної хеш-функції зображення особливо важлива, коли потребує хешування та обробки велика кількість зображень. Це, наприклад, випадок перевірки авторського права на великій базі даних. Також швидкодія критична у високонавантажених системах та системах з обмеженими ресурсами. Для порівняльного аналізу використано набір різнотипних зображень. У дослідженні замірюється час безпосереднього виконання алгоритмів та не враховується час завантаження зображення. Результати наведені у таблиці 4.2 та візуалізовані на рис. 4.1. Для тестування використовується конфігурація ЕОМ, вказана в таблиці 4.1

Таблиця 4.1 – конфігурація ЕОМ, на якій проводяться тести

CPU	Процессор Intel® Core™ i5-6400
RAM	16GB
SSD	Samsung EVO 860 500GB 2.5'' SATA III
OS	Windows 10 Pro x64

Таблиця 4.2 – Тестування модифікованого алгоритму та алгоритму pHash на швидкодію

	pHash DCT-II	Розроблений алгоритм
Сумарний час (мілісекунди)	3644	232
Середній час обробки одного зображення	36.4	2.32

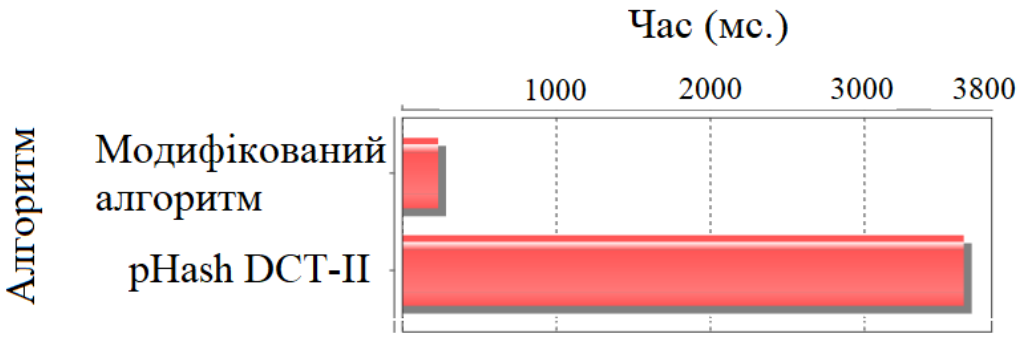

Рисунок 4.1 – Гістаграма порівняння швидкодії алгоритмів

З результатів видно, що модифікований алгоритм має набагато швидший час виконання, порівнюючи з алгоритмом pHash. Причиною є те, що pHash використовує двомірне обчислення ДКП-2 у своїй основі. Розроблений алгоритм виконує прості обчислення, такі як обчислення середніх значень невеликих векторів.

4.3 Розподіл значень немодифікованих зображень

Розподіл значень може бути використаний для вимірювання та оцінки дискримінаційних можливостей перцептивної хеш-функції. Порівнюючи два різні зображення, ідеальна перцептивна хеш-функція завжди повинна давати відстань (в даному випадку, оцінку подібності BER) 0,5. Цікавим питанням є те, чи залежить розподіл балів від використаних зображень. При використанні дуже подібних зображень для хеш-функції сприйнятливого зображення може бути важче досягти «ідеальної» відстані 0,5 для кожного порівняння. Тисяча зображень, серед яких зображені засніжені гори, можна розглядати як подібний набір подібних зображень. Тому розподіл між балами визначався за

допомогою двох різних наборів зображень. Першим набором був набір випадкових зображень (набір 1 в таблиці), тоді як другий набір - набір зображень качки (набір 2 в таблиці). Кожна хеш-функція перцептивного зображення повинна була обчислити $100^2 = 1000$ хеш-кодів для кожного набору зображень. Результати вказані в таблиці 4.3.

Таблиця 4.3 – Статистичні результати розподілу значень немодифікованих зображень

	pHash DCT-II. Набір 1	Розроблений алгоритм. Набір 1	pHash DCT-II. Набір 2	Розроблений алгоритм. Набір 2
Середня відстань	0.501	0.531	0.496	0.484
Максимальна відстань	0.688	0.702	0.750	0.739
Мінімальна відстань	0.250	0.200	0.219	0.194

Згідно з результатами, у середньому pHash дає кращий розподіл, близький до 0.5. Причиною таких гарних результатів є використання ДКП-2. Розроблений алгоритм не використовує математичних перетворень для

визначення низьких частот та складової частини зображення, тому результати є дещо гіршими.

4.4 Зміна розміру зображеннь

Таблиця 4.4 узагальнює результати розподілу значень при зміні розміру зображення. Зображення були змінені шляхом зміни ширини до 1024 пікселів. Висота відрегульована пропорційно. Використовується бікубічна інтерполяція для кращих результатів.

Для тестування використовується набір 1 довільних зображень. Для оригінального зображення обчислюється хеш-код. Потім зображення модифікується шляхом зменшення його розміру. Обчислюється хеш-код модифікованого зображення. Обидва хеш-коди порівняні між собою за допомогою відстані Гемінга, на основі якого обчислюється значення BER, що вказані у таблиці 4.4.

Таблиця 4.4 – Тестування запропонованого алгоритму та алгоритму pHash при зміні ширини зображення до 1024 пікселів. Застосована кубічна інтерполяція.

	pHash DCT-II	Модифікований алгоритм
Середня відстань	0.076	0.012
Максимальна відстань	0.219	0.039
Мінімальна відстань	0.000	0.000

Обидва алгоритми є однаково інваріантними до зміни розміру зображення. Причиною цього є те, що обидві алгоритми не використовую деталізовану складову зображення його ідентифікації. Навпаки, самі алгоритми зменшують зображення у розмірі для позбавлення від високих частот та простоти обчислень.

4.5 JPEG стиснення зображень

Зображення модифіковані шляхом JPEG компрессії. Параметр якості JPEG - 80. Крім того, досліджено налаштування якості JPEG відносно надійності перцептивних хеш-функцій зображення. Параметр якості поступово змінювали від 100 до 0. Для кожного значення параметра якості обчислені оцінки відстані даних зображень.

Для тестування використовується набір 1 довільних зображень. Для оригінального зображення обчислюється хеш-код. Потім зображення модифікується шляхом компресії JPEG. Обчислюється хеш-код модифікованого зображення. Обидва хеш-коди порівняні між собою за допомогою відстані Гемінга, на основі якого обчислюється значення BER, що вказані у таблиці 4.5.

Таблиця 4.5– Тестування запропонованого алгоритму та алгоритму pHash при JPEG компресії зображення з параметром якості 80

	pHash DCT-II	Модифікований алгоритм
Середня відстань	0.001	0.001
Максимальна відстань	0.031	0.008
Мінімальна відстань	0.000	0.000

Обидва алгоритми є однаково інваріантними до стиснення зображення алгоритмом JPEG з параметрами якості від 10 до 90. Причиною цього є те, що обидві алгоритми не використовую деталізовану складову зображення його ідентифікації. Навпаки, самі алгоритми зменшують зображення у розмірі для позбавлення від високих частот та простоти обчислень.

4.6 Обертання зображеннь

Зображення модифіковані шляхом їх обертання. Для тестування використовується набір 1 довільних зображень. Для оригінального зображення обчислюється хеш-код. Потім зображення модифікується шляхом обертання на 5°. Обчислюється хеш-код модифікованого зображення. Обидва хеш-коди порівняні між собою за допомогою відстані Гемінга, на основі якого обчислюється значення BER, що вказані у таблиці 4.6.

Таблиця 4.6 – Тестування запропонованого алгоритму та алгоритму pHash при обертанні зображення на 5°

	pHash DCT-II	Розроблений алгоритм

Середня відстань	0.235	0.103
Максимальна відстань	0.400	0.245
Мінімальна відстань	0.092	0.067

Згідно з результатами, алгоритм pHash у седедньому дає значення, на основі яких не недоцільно стверджувати схожість зображень. Розроблений алгоритм, у свою чергу, дає кращі результати.

4.7 Томографічні знімки

Метою даного тесту є тестування є перевірка результатів порівняння модифікованих зображень з оригінальними, використовуючи широкий ряд модифікацій.. Для детального тестування запропонованого алгоритму використані чотири зображення:

1) Томографічний знімок легенів людини;
2) Томографічний знімок мозку людини;
3) Зображення інтер’єру кімнати;
4) Зображення дорожнього трафіку у час пік.

Перше та друге зображення є томографічними знімками, чорно-білими по визначенню. Третє та четверте зображення є кольоровими зображеннями розміром 2874 x 2260 та у форматі JPG, що випадково вибрані із першого набору минулих тестів. Алгоритм використовує конфігурацію алгоритму, що вказана у таблиці 4.6.

Таблиця 4.6 – Налаштування запропонованого алгоритму

Метод зменшення розміру зображення	Бікубічна інтерполяція
Розмір зображення зменшений до	256x256
Довжина хеш-коду у бітах, N	64
Метод зведення у градації сірого	Average

Для кожного зображення створено 25 модифікованих наступними методами:

1) Стиснення: JPG Q = 10; JPG Q = 40; JPG Q = 90;
2) Операція фільтрування: Гаусова фільтрація 3 × 3; збільшення різкості зображення 3×3; медіанна фільтрація 3×3;
3) Геометричні спотворення: масштабування 0.5; масштабування 2.0; обрізання 2%; обрізання 10%; обрізання 20%; випадкове видалення 5% площі; обертання 2°, 5° , 8° , 10°, 15°, 30° та 90°;
4) Зашумлення: Гаусове зашумлення 0.01; Salt&Pepper (SAP) зашумлення 0.02; Спекл зашумлення 0.01;
5) Збільшення яскравості.

Результати представлены у значеннях BER та наведені в таблиці 4.7:

Таблиця 4.7 – BER чотирьох зображень. Чотири перших рядки – оригінальні зображення, наступні 25 рядків – модифікації оригінальних зображень

Зображення та модифікації	Зображення 1	Зображення 2	Зображення 3	Зображення 4
Зображення 1	0.000	0.394	0.550	0.461
Зображення 2	0.337	0.000	0.531	0.457

Зображення 3	0.547	0.564	0.000	0.480
Зображення 4	0.535	0.573	0.480	0.000
JPG Q = 10	0.004	0.004	0.008	0.004

JPG Q = 40	0.003	0.002	0.004	0.000
JPG Q = 90	0.000	0.000	0.000	0.000
Гаусовий фільтр 3x3	0.002	0.000	0.004	0.000
Збільшення різкості зображеня 3x3	0.019	0.008	0.023	0.004
Медіанна фільтрація 3x3	0.000	0.000	0.000	0.000
Масштабування 0.5	0.000	0.000	0.000	0.000
Масштабування 2.0	0.001	0.000	0.004	0.000
Обрізання 2%	0.063	0.035	0.074	0.027
Обрізання 10%	0.201	0.175	0.234	0.121
Обрізання 20%	0.344	0.221	0.363	0.219
Випадкове видалення 10%	0.024	0.020	0.027	0.016
Обертання 2°	0.037	0.041	0.048	0.051
Обертання 5°	0.079	0.080	0.088	0.091

Обертання 8°	0.157	0.161	0.172	0.175
Обертання 10°	0.193	0.198	0.226	0.215
Обертання 15°	0.297	0.303	0.328	0.313
Обертання 30°	0.435	0.441	0.468	0.471
Обертання 90°	0.571	0.569	0.585	0.595

Гаусове зашумлення 0.01	0.003	0.003	0.004	0.004
SAP зашумлення 0.02	0.002	0.002	0.004	0.002
Спекл зашумлення 0.01	0.002	0.003	0.004	0.004
Збільшення яскравості	0.026	0.019	0.023	0.008

Згідно до результатів, алгоритм забезпечує інваріантність до наступних модифікацій зображень:

1) Стиснення JPEG;
2) Масштабування;
3) Збільшення різкості;
4) Гаусової, медіанної фільтрації;

5) Зашумлення зображення (наприклад, Гаусове зашумлення, SAP та Спекл зашумлення);

6) Збільшення яскравості зображення.

Також, розроблений алгоритм забезпечує часткову інваріантність до обзірання зображення, випадкового видалення вмісту зображення, а також – до обертання зображення. Згідно до таблиці 4.7, алгоритм спроможний виявити схожість зображень, обернених до 8°. При обертанні зображення на 10° маємо коефіцієнти BER, приближені до 0.2. Такі значення коефіцієнтів не є інформативними для зіставлення зображень. Для томографічних знімків результати дещо кращі (приблизо на 5%). Це пояснюється їх однотипністю та більшою кількістю однотонних (майже однотонних) областей, обертання яких впиває на результат меншою мірою.

Висновки до розділу

Проведено ряд тестів та досліджень запропонованого алгоритму. Для тестів використані набори зображень у високій роздільній здатності. Наведені результати порівняння запропонованого алгоритму з алгоритмом pHash. Проведений ряд тестів на інваріантність запропонованого алгоритму до різних модифікацій різних зображень, в тому числі для томографічних знімків мозку та легенів людини. Досліджено, що розроблений алгоритм інваріантний до стиснення, масштабування, основних алгоритмів фільтрації та зашумлення зображення. Також, розроблений алгоритм спроможний порівняти зображення, що модифіковані шляхом обертання до 8° відносно оригіналу. Для томографічних знімків цей результат декілька кращий. Це є результатом того, що томографічні знімки містить більшу кількість однотонних (майже однотонних) областей. Розбиття зображення на блоки дозволяє забезпечити кращу інваріантність до обертання таких областей.

Також, даний алгоритм є швидким у плані часу виконання. У середньому, алгоритм виконується на 10-11 разів швидше алгоритму pHash та має приблизно однакові часові показники з простими алгоритмами, такі як Average Hash та Difference Hash. Однак, алгоритм показує гірші дискримінаційні здібності, у порівнянні з алгоритмом pHash. Це є результатом того, що алгоритм pHash виконує математичні перетворення для визначення ключової складової зображення, для якої обчислюється значення хеш-коду. Розроблений алгоритм не використовує ці переваги.

ВИСНОВКИ

У магістерській дисертації розглянуто алгоритми порівняння зображень. Умовно такі алгоритми ділять на перцептивні алгоритми обчислення хеш-коду зображення та алгоритми визначення ознак та ключових точок зображення.

Визначено, що перцептивні хеш-алгоритми є достатньо простими у реалізації та забезпечують інваріантність до більшості модифікацій зображень, окрім їх обертання, обрізання та модифікації вмісту.

Визначені основні характеристики, які повинен мати перцептивний хеш-алгоритм для порівняння медичних томографічних зображень. Однією важливою властивістю є інваріантність до обертання зображень на невеликий кут повороту (до 5°).

Запропоновано метод розбиття зображення на блоки для обчислення хеш-коду для томографічних зображень.

Запропоновано алгоритм, який є модифікацією алгоритму Average Hash. Ідея блочної обробки зображень зоснована на використанні алгоритму BTC.

Проведено ряд тестів та досліджень запропонованого алгоритму. Для тестів використані набори зображень у високій роздільній здатності.

Наведено результати порівняння запропонованого алгоритму з алгоритмом pHash. Досліджено, що розроблений алгоритм інваріантний до стиснення, масштабування, основних алгоритмів фільтрації та зашумлення зображення. Також, запропонований алгоритм спроможний порівняти зображення, що модифіковані шляхом обертання до 8° відносно оригіналу.

Доведено, що запропонований алгоритм є швидким за часом виконання. У середньому, алгоритм виконується в 10-11 разів швидше алгоритму pHash

та має приблизно однакові часові показники з простими алгоритмами, такі як Average Hash та Difference Hash.

Однак, алгоритм показує гірші дискримінаційні здібності, у порівнянні з алгоритмом pHash. Дискримінаційна властивість може бути критичною для порівняння різнотипних зображень та модифікації великої площі зображень. Однак, це не є критичним для порівняння однотипних томографічних зображень, над якими не проводять модифікації зміни їх вмісту. Тобто, розроблений алгоритм дозволить дати порівняльну оцінку томографічним знімкам, дозволить лікарям ефективніше відслідковувати процес лікування пацієнтів та зміни в його організмі за допомогою томографічних знімків та їх порівняння.

Швидкодія алгоритму дозволяє його використовувати у навантажених системах. Також, даний алгоритм може використовуватися в інших областях, для яких може бути корисним використання переваг запропонованого алгоритму.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Coskun, B., Memon, N: Confusion/diffusion capabilities of some robust hash functions. In Proceedings of the Conference on Information Sciences and Systems (CISS). IEEE, 2006. - pp. 1188–1193
2. Cox, I.J., Miller, M.L., Bloom, J.A.: Digital Watermarking. Morgan Kaufmann, 2002, ISBN 1558607145
3. Schneider, M., Chang, S.F.: A robust content based digital signature for image authentication. In: Proc. of International Conference on Image Processing (ICIP1996), 1996. - pp. 227–230
4. Hamming, R.W.: Error detecting and error correcting codes. The Bell System Technical Journal, 1950
5. Swaminathan, A., Mao, Y., Wu, M.: Robust and secure imagehashing. IEEE Transactions on Information Forensics and Security, 2006 – pp. 215–230.
6. Drakos, N. Moore, R.: Definition of DCT, 2010.
7. Bernard Vukelić, Miroslav Bača: Comparison of RADIAL variance based and Maar-Hildreth operator perceptual image hash functions on biometric templates, 2014
8. D. Marr, E. Hildred: Theory of edge detection, 1980
9. David G. Lowe: Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision, vol.50, No. 2, 2004. - pp.91-110
10. Herbert Bay, Tinne Tuytelaars, Luc Van Gool: Speeded-up robust features (SURF). Computer vision and image understanding, vol.110, No.3, 2008. - pp. 346-359
11. Y. Cong, X. Chen, Y. Li, Research on the SIFT Algorithm in Image Matching, AMM, vol. 121-126, 2011

12. X. Qiao: Research on the Algorithm of Image Matching Based on Improved SIFT, AMM, vol. 686, 2014 - pp. 348-353
13. D. Nister, H. Stewenius: Scalable recognition with a vocabulary tree, Proc. International Conference on Computer Vision and Pattern Recognition, Vol. 2, pp.2161-2168. - 2006
14. Deepak Geetha Viswanathan: Features from Accelerated Segment Test (FAST), 2011
15. Menezes, A.J., Vanstone, S.A., Oorschot, P.C.V.: Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA, 1996, ISBN 0849385237
16. Mihcak, M., Venkatesan, R.: New iterative geometric methods for robust perceptual image hashing. In Revised Papers from the ACM CCS-8 Workshop on Security and Privacy in Digital Rights Management, vol.2200 of Lecture Notes in Computer Science, 2001
17. Monga, V.: Perceptually Based Methods for Robust Image Hashing. PhD thesis, University of Texas, 2005
18. Caldelli, R., Vogel, T., Dittmann, J., Thiemert, S., Solachidis, V., Voloshynovskiy, S., Deguillaume, F., Pun, T., Minguillon, J., Megias, D., Schmucker, M., Steinebach, M.: First summary reporton authentication. Tech. Rep. D.WVL.6, ECRYPT, 2005.
19. Schneider, M. Chang, S.F.: A robust content based digital signature for image authentication. In Proceedings of the International Conference on Image Processing (ICIP), vol. 3, IEEE, 1996. - pp. 227–230
20. Cano, P.: Content-Based Audio Search: from Fingerprinting to Semantic Audio Retrieval. PhD thesis, Universitat Pompeu Fabra, 2007

21. Haitsma, J., Kalker, T., Oostveen, J.: An efficient database search strategy for audio fingerprinting. In Proceedings of the Workshop on Multimedia Signal Processing (MMSP). IEEE, 2002. - pp. 178–181
22. Dhawan, Atam P: Medical Image Analysis. Hoboken, NJ: Wiley-Interscience, 2003
23. Delp, E., Mitchell, O.: Image Compression Using Block Truncation Coding. IEEE Transactions on Communications, 1979
24. Campbell, G., Defanti, T. A., Frederiksen, J. Joyce, S. A. Leske, L. A.: Two bit/pixel full color encoding. Proceedings of the 13th annual conference on Computer graphics and interactive techniques - SIGGRAPH '86, 1986. - p. 215
25. https://visualhunt.com – відкрита база даних зображень
26. T. Blu, P. Thevenaz, and M. Usner, “Linear Interpolation Revitalized,” IEEE Transactions on Image Processing, Vol. 13, No. 5, May 2004
27. https://docs.gimp.org/2.6/en/gimp-tool-desaturate.html – документація Gimp
28. https://storage.googleapis.com/openimages/web/index.html – відкрита база даних зображень